С.Д. ПАРАЩУК

ПРАКТИКУМ
ІЗ ПРОЦЕДУРНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ (МОВА С)

НАВЧАЛЬНИЙ ПОСІБНИК ДЛЯ СТУДЕНТІВ ВИЩИХ НАВЧАЛЬНИХ ЗАКЛАДІВ

С.Д.Паращук

ПРАКТИКУМ
іЗ ПРОЦЕДУРНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ
(мова С)

Навчальний посібник для студентів вищих навчальних закладів

Кіровоград 2016

ББК 32.973.2-018
П18
УДК 519.688
ISBN 978-966-2466-54-5

Паращук С.Д.
Практикум із процедурно-орієнтованого програмування (мова С): Навчальний посібник. – Кіровоград: ФО-П Александрова М.В., 2016. – 220 с.

Рецензенти:
Д.Б.Буй, доктор фізико-математичних , професор, завідувач кафедри інформатики Кіровоградського державного педагогічного університету імені Володимира Винниченка (професор кафедри теорії і технології програмування факультету кібернетики Київського національного університету імені Тараса Шевченка)
О.В.Авраменко, доктор фізико-математичних наук, професор, завідувач кафедри прикладної математики, статистики та економіки Кіровоградського державного педагогічного університету імені Володимира Винниченка

Рекомендовано рішенням Методичної ради Кіровоградського державного педагогічного університету імені Володимира Винниченка (протокол № 7 від 13 квітня 2016 року)

Рекомендовано рішенням Вченої ради Кіровоградського державного педагогічного університету імені Володимира Винниченка (протокол № 10 від 25 квітня 2016 року)

Мета посібника – розвиток навичок алгоритмізації та програмування мовою С, формування вмінь розробляти алгоритми та створювати, редагувати і налагоджувати програми.

У посібнику подані матеріали для проведення практичних занять, що охоплюють усі розділи з основ програмування. До кожного заняття подані короткі теоретичні відомості, зразки створення алгоритмів і програм, завдання для індивідуальної роботи.

Посібник призначений для студентів, які навчаються за галуззю знань «Інформаційні технології» та «Освіта» (за спеціальністю або спеціалізацією «Інформатика»)

ISBN 978-966-2466-54-5

Зміст

Передмова

Мова С є простою для вивчення і має всі засоби, що притаманні сучасним мовам програмування. Вона придатна для структурного програмування, орієнтована на обробку різного виду інформації. Окрім того, багато сучасних мов програмування є С-подібними. Тому при вивченні основ програмування має сенс використовувати цю мову. Однак, зауважимо, що вона не є строго регламентованою і в цьому плані мова С програє мові Паскаль, як мова для навчання. Тому при вивченні основ програмування на базі мови С, викладач повинен формувати у студентів культуру програмування, яка орієнтована на сучасні мови програмування.

Навчальний посібник містить повний цикл практичних занять із основ програмування, орієнтованих на інтегровані середовища програмування, які підтримують мову С++. Тематика занять охоплює наступні розділи програмування: оператори, підпрограми та рекурсія, структуровані типи даних, сортування і пошук, динамічне програмування і динамічні структури даних, заголовкові файли.

Кожне заняття розпочинається з коротких теоретичних відомостей. Тут зібрані основні факти, що необхідні для написання програм. Однак студенти повинні володіти теоретичним матеріалом у повному обсязі при складанні звітів із виконаних завдань. Далі у роботі містяться приклади розв’язання задач. Наводиться їхній аналіз і пропонується алгоритм вирішення задач. Приклад завершується текстом програми, яка є консольним додатком Visual Studio 2010. Ці приклади можуть слугувати зразками для виконання індивідуальних завдань. У текстах програм наводиться достатня кількість коментарів, які пояснюють призначення відповідних конструкцій і розкривають алгоритм розв’язання задач.

Кожне практичне заняття містить завдання для індивідуального виконання. Вони розбиті на два рівні – основний та підвищений. Завдання основного рівня є нескладними і в основному охоплюють клас базових задач із відповідної теми. Завдання підвищеного рівня розраховані на студентів з високим рівнем знань. У деяких темах також є додаткові задачі. Їх розв’язання, зазвичай, вимагає додаткових знань і розробки ефективних алгоритмів.

Навчальний посібник орієнтований на різні форми організації роботи. Деяку частину індивідуальних завдань можна внести до обов’язкового обсягу, яка контролюється викладачем під час аудиторних занять. Іншу частину завдань можна винести на самостійне опрацювання. Серед додаткових задач можна вибрати такі, які вимагають розробки індивідуального проекту та завершеного програмного продукту. Ці задачі можна рекомендувати як початковий етап майбутнього курсового проекту.

Практичне заняття № 1

Тема: Лінійні програми

Студент повинен знати: структуру програми, прості числові типи даних та операції над цими даними, вирази, синтаксис і семантику операції присвоювання, математичні функції для числових типів даних.

Теоретичні відомості

Основними складовими будь-якої С-програми є функції. Вони діляться на бібліотечні та користувацькі. Перші зберігаються в бібліотеці системи програмування, а другі – створюються програмістом у даній програмі. Проста програма мови С має вигляд

директиви_препроцесора
int main(void)
{
визначення_об'єктів;
оператори;
return 0;
}

У програмі обов'язково повинна бути функція **main** (головна) - з неї починається виконання програми, а завершення виконання **main** призводить до завершення роботи програми. Функція **main** є користувацькою. Оператор

return 0;

завершує роботу функції **main** та програми в цілому і передає операційній системі значення 0, яке означає коректне завершення роботи програми. У тілі функції **main** зазвичай оголошуються та означаються різні об'єкти. Найчастіше – це змінні або константи. Оператори призначені виконувати певні дії на кожному кроці роботи програми.

Директива препроцесора

#include <ім'я_заголовкового_файлу>

підключає до програми файли зі стандартними бібліотечними функціями, які можна використовувати у програмі. Зокрема, директива **#include <stdio.h>** використовується для підключення до програми стандартних засобів введення-виведення. Мова С немає власних операторів для введення-виведення даних. Тому до програми підключається файл **stdio.h** і його функції використовуються для введення-виведення. Найчастіше використовують функції **printf** та **scanf** для форматного введення-виведення даних.

Прототип функції **printf** має вигляд

int printf(<рядок_формату>, <список_виразів_виведення>);

Вона перетворює значення виразів, заданих у списку виразів виведення, відповідно до специфікації, записаних у рядку формату, і передає результат перетворення у стандартний вихідний потік **stdout**, який здебільшого пов'язаний з виведенням на екран. Функція повертає кількість виведених символів, якщо вона успішно завершилась, інакше – повертає від'ємне число.

Першим параметром функції **printf** є **рядок_формату**. Це рядок символів, який може складатися із трьох груп символів: символи кодової таблиці, керуючі символи, специфікації формату. Символи кодової таблиці використовуються для написання тексту, який виводиться без змін. Керуючі символи записуються разом зі знаком \. Наприклад, **\n** – перехід на навий рядок, **\t** – горизонтальна табуляція. Специфікація формату визначає форму виведення значення відповідного виразу. Її синтаксис має вигляд

%[прапорці][ширина][.точність][модифікатор]символ_специфікації

Кожна специфікація формату починається з % і закінчується символом_специфікації, які є обов'язковими. Решта параметрів записані в квадратних дужках, тобто вони є не обов'язковими. Кількість специфікацій у **рядку_формату** повинна співпадати з кількістю

виразів у **списку_виразів виведення** (хоч це не обов'язково). Відповідність між специфікаціями та виразами – позиційна: перша специфікація визначає формат виведення першого виразу, друга – другого і т.д.

Прототип функції **scanf** має вигляд

int scanf(<рядок_формату>, <список_адрес_виведення>);

Вона використовується для форматного введення даних з клавіатури і специфікації форматів в неї аналогічні як для функції printf.

Часто для виведення рядків використовується функція **puts**.

Під час оголошення змінної треба вказувати її тип. Всі типи мови С можна розділити на три групи: *скалярні* (або *прості*), *структуровані* (або *складені*), тип «функція». Скалярними є арифметичні типи, переліки, вказівники. Арифметичні типи ще діляться на *цілочислові* та *дійсні*. Кожний арифметичний тип має ім'я, яким перш за все визначається діапазон можливих значень для даних цього типу.

Крім цього, типи даних поділяються на *базові* та *похідні*. До базових типів відносяться арифметичні типи та тип **void**. Базові типи вбудовані в систему програмування. Тип **void** означає порожній або невизначений тип. Його часто використовуються в оголошеннях функцій, щоб вказати, що функція не повертає ніякого значення в точку виклику або вона немає параметрів. У таблиці 1.1 подана інформація про арифметичні типи даних мови С.

Таблиця 1.1. Арифметичні типи даних.

Назва типу даних	**Повна назва типу даних**	**Розмір (у байтах)**	**Діапазон значень**
Цілочислові типи			
unsigned char	**unsigned char**	1	0…255
char	**signed char**	1	-128…127
unsigned short	**unsigned short int**	2	0…65535
short	**signed short int**	2	-32768…32767
unsigned int	**unsigned int**	2	0…65535
int	**signed int**	2	-32768…32767
unsigned long	**unsigned long int**	4	0…4294967295
long	**signed long int**	4	-2147483648…2147483647
long long	**signed long long int**	8	-9223372036854775808… 9223372036854775807
Дійсні типи			
float	**float**	4	3.4E-38…3.4E+38
double	**double**	8	1.7E-308…1.7E+308
long double	**long double**	10	3.4E-4932…1.1E+4932

Основними цілочисловими типами є **char** та **int**. До них застосовують модифікатори **short**, **long**, **signed**, **unsigned** та отримують інші цілочислові типи. Модифікатор **signed** (знаковий) зазвичай опускають в назві типу. Також дозволяється опускати **int**, там де можливо.

Змінні у програмі оголошуються за синтаксисом

<тип> <ім'я_змінної>;

або **<тип> <список_змінних>;**

Під час означення змінної її можна ініціалізувати початковим значенням, яке є значенням виразу

<тип> <ім'я_змінної> = <вираз>;

Вирази мови С мають складну структуру. Вони утворюються із операндів за допомогою операцій. Усі вирази мови С поділяються на три види: l-, r- та f-вирази. l- та r-вирази – це конструкції, що за синтаксисом можуть бути відповідно лівим і правим операндами в операції присвоювання. f-виразами називають конструкції, які використовуються як імена функцій.

Значеннями l-виразів є адреси об'єктів певного типу. Найпростішим прикладом l-виразу є змінна. Синтаксис l-виразів можна записати так.

< l-вираз > ::= < змінна-стандартного-типу >| < індексований-вираз >|
< вибір-компонентів >|< змінна-вказівник >|
< розіменування-вказівника >|(< l-вираз >)

r-вирази є найчисленнішими за кількістю операцій і складністю. Їх можна класифікувати за наступним означенням.

< r-вираз >::=< первинний-вираз > | <вираз-присвоювання > |
< постфіксний-вираз > | < унарний-вираз > | < бінарний-вираз > |
< логічний-вираз > | < умовний-вираз > | < послідовний-вираз > |
< константний-вираз >

З первинних виразів за допомогою операцій будуються всі решта:

< первинний-вираз >::=< l-вираз > | <константа > | (< вираз >)

Вирази присвоювання оновлюють значення змінних:

<вираз-присвоювання>::=<l-вираз><операція-присвоювання><r-вираз>

Ми не будемо далі уточнювати різні варіанти виразів. Найпростішими прикладами виразів є змінна, константа, виклик функції. Зазвичай з них утворюють складніші вирази. У виразі використовують круглі дужки, щоб змінити порядок виконання операцій. У таблиці 1.2 міститься список усіх операцій мови С та їхній пріоритет.

Таблиця 1.2. Операції мови С.

Пріоритет	Знак операції	Тип операції	Асоціаттивність
1	() виклик функції [] індексація . взяття поля -> розіменування поля	Бінарна Бінарна Бінарна	Ліва
2	~ побітове заперечення ! заперечення * розіменування & взяття адреси ++ збільшення (інкремент) -- зменшення (декремент) +, - (тип) зведення типу sizeof розмір типу	Унарні	Права
3	* множення / ділення % залишок	Мультипліка тивні	Ліва
4	+, -	Адитивні	Ліва
5	<< зсув ліворуч >> зсув праворуч	Побітові	Ліва
6	<, >, <=, >=	Відношення	Ліва
7	== рівність != нерівність	Відношення	Ліва
8	& кон'юнкція	Побітова	Ліва
9	^ додавання за mod 2	Побітова	Ліва
10	| диз'юнкція	Побітова	Ліва
11	&& кон'юнкція	Логічна	Ліва
12	|| диз'юнкція	Логічна	Ліва
13	?: умовна		Ліва
14	=, *=, /=, %=, +=, -=, &=, |=, >>=, <<=, ^=	Присвоювання	Права
15	, послідовне обчислення		Ліва

Лінійними називаються програми, які побудовані лише з простих операторів на основі конструкції слідування. Простими операторами мови С є: порожній оператор, оператор виразу, оператори goto, break, continue, return.

Синтаксис оператора виразу

<вираз>;

У лінійних програмах зазвичай використовуються тільки такі оператори. Прикладом оператора виразу є присвоювання, найпростіший варіант якого має вигляд

<ім'я_змінної>=<вираз>;

Під час обчислення арифметичних виразів часто використовують стандартні математичні функції, які оголошені в стандартному заголовковому файлі math.h. Цей файл можна підключити до програми через директиву **#include** <math.h>. Основні математичні функції файлу math.h вказані в таблиці 1.3.

Таблиця 1.3. Математичні функції.

Виклик функції	**Прототип функції**	**Опис функції**
abs(x)	int abs(int x);	|x| - абсолютна величина x. Повертає модуль цілого аргументу x.
fabs(x)	double fabs(double x);	|x| - абсолютна величина x. Повертає модуль дійсного аргументу x.
labs(x)	long abs(long x);	|x| - абсолютна величина x. Повертає модуль довгого цілого аргументу x.
exp(x)	double exp(double x);	e^x - експонента x. Повертає значення e^x.
log(x)	double log(double x);	ln x – логарифм натуральний x (x>0). Повертає значення натурального логарифму ln x.
log10(x)	double log10(double x);	lg x – логарифм десятковий x (x>0). Повертає значення десяткового логарифму lg x.
pow(x, y)	double pow(double x, double y);	x^y – піднесення x до дійсного степеня y (помилка, якщо: 1) x=0, а y≤0; 2) x<0, а y – не є цілим). Повертає значення x^y.
pow10(p)	double pow10(int p);	10^p – піднесення 10 до цілого степеня p. Повертає значення 10^p.
sqrt(x)	double sqrt(double x);	√x – корінь квадратний з x (x≥0). Повертає значення √x.
sin(x)	double sin(double x);	sin x – синус x (значення x задається в радіанах). Повертає значення sin x.
cos(x)	double cos(double x);	cos x – косинус x (значення x задається в радіанах). Повертає значення cos x.
tan(x)	double tan(double x);	tg x – тангенс x (значення x задається в радіанах). Повертає значення tg x.
asin(x)	double asin(double x);	arcsin x – арксинус x ($x \in [-1, 1]$). Повертає значення arcsin x у діапазоні [-π/2, π/2].
acos(x)	double acos(double x);	arccos x – арккосинус x ($x \in [-1, 1]$). Повертає значення arccos x у діапазоні [0, π].
atan(x)	double atan(double x);	arctg x – арктангенс x. Повертає значення arctg x у діапазоні [-π/2, π/2].
atan2(x,y)	double atan2(double x, double y);	arctg x/y – арктангенс x/y. Повертає значення arctg x/y у діапазоні [-π, π].
sinh(x)	double sinh(double x);	sh x – гіперболічний синус x. Повертає значення sh x.
cosh(x)	double cosh(double x);	ch x – гіперболічний косинус x.

		Повертає значення ch x.
tanh(x)	double tanh(double x);	th x – гіперболічний тангенс x. Повертає значення th x.
ceil(x)	double ceil(double x);	найменше ціле ≥ x. Повертає значення в формі double.
floor(x)	double floor(double x);	найбільше ціле ≤ x. Повертає значення в формі double.

Для обчислення інших арифметичних функцій можна скористатися наступними тотожностями:

$ctg\ x = \frac{\cos x}{\sin x}$; $arcctg\ x = arctg\frac{1}{x}$, $x \neq 0$; $\log_a x = \frac{\ln x}{\ln a}$, $a > 0, a \neq 1, x > 0$;

$a^x = e^{x \cdot \ln a}$, $a > 0, a \neq 1$; $x^y = e^{y \cdot \ln x}$, $x > 0, x \neq 1$.

Приклад 1

Дано площу круга. Знайти сторону правильного трикутника, вписаного в коло.

Аналіз задачі

Площа круга обчислюється за формулою $S = \pi * R^2$, а сторона правильного трикутника, вписаного в коло, дорівнює $a = R * \sqrt{3}$. Отже, з першої формули треба обчислити $R = \sqrt{S/\pi}$ і підставити результат у другу формулу.

Алгоритм очевидний і наведений в тексті програми.

```
// Підлючення бібліотечних файлів
#include "stdafx.h"
#include "math.h"
#include "windows.h"
#include "locale.h"
#define PI 3.14159              // Визначення числа π як константи

int _tmain(int argc, _TCHAR* argv[])        // Заголовок аналога функції main
{       setlocale (0,"");               // Пітримка виведення російських букв на екран
        double s,r,a;   // Оглошення змінних дійсного типу: s – площа, r – радіус, a –
                                                                    сторона трикутника
        printf("Введіть площу круга s=");           // Виведення підказки на екран
                                                    // Виклик функції printf
        scanf("%lf", &s);               // Введення зачення площі з клавіатури
                                        // Виклик функції scanf
        r=sqrt(s/PI);                   // Обчислення r. Функція sqrt із файлу math.h
        a=r*sqrt(3.);                   // Обчислення a
        printf("Сторона трикутника =%lf\n", a);    // Виведення a
        system("pause");                // Затримка програми
        return 0;
}
```

```
Введіть площу круга s=55,67
Сторона трикутника =7,291158
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 2

Дано додатне число x. Обчислити $y = \frac{(5x+3)^{2x+1} \cdot \sqrt[5]{\sin^2(3x^2 - 4x + 5)}}{3^{4x+3}}$.

Аналіз задачі

Алгоритм розв'язання задачі простий: ввести x, обчислити y, вивести y. Основна складність в тому, щоб правильно записати мовою С формулу для обчислення y. Для цього потрібно скористатися стандартними математичними функціями із заголовкового файлу **math.h**, які наведені в таблиці 1.3, та основними математичними тотожностями.

```
#include "stdafx.h"
#include "windows.h"
#include "locale.h"
#include "math.h"                    // Підключення бібліотеки математичних функцій

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (0,"");
        double x, y;                 // Оголошення змінних x та y
        printf("Введіть додатне число x\n");
        scanf("%lf", &x);            // Введення x з клавіатури
        // Обчислення y
        y=pow(5*x+3, 2*x+1)*pow(sin(3*x*x-4*x+5)*sin(3*x*x-4*x+5),1./5)/exp((4*x+3)
                                                                    *log(3.));
        printf("Значення y = %lf\n", y);     // Виведення y
        system("pause");
        return 0;
}
```

```
Введіть додатне число x
0,45
Значення y = 0,098731
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Задачі

1. Дано додатні числа a, b. Обчислити
$$y = \sqrt[5]{\log_a b} \cdot \sin a^b .$$
2. Дано координати вершин трикутника: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Обчислити периметр і площу трикутника.
3. Дано x. Обчислити $2x^4-3x^3+4x^2-5x+6$ за найменшу кількість операцій.
4. Обчислити дробову частину середнього геометричного трьох даних додатних чисел.

☺ Індивідуальні завдання

Основний рівень

1. Дано сторони трикутника. Обчислити його площу.
2. Дано висоту та радіус основи конуса. Обчислити його об'єм.
3. Дано площу повної поверхні та радіус основи циліндра. Обчислити його висоту.
4. Дано дві сторони трикутника та кут між ними (в градусах). Обчислити площу трикутника.
5. Дано сторони трикутника. Знайти його висоти.
6. Дано сторону ромба та гострий кут між сторонами (в градусах). Знайти площу ромба.
7. Дано основи та висоту трапеції. Знайти її площу.
8. Дано висоту та сторону основи правильної чотирикутної піраміди. Обчислити її об'єм.
9. Дано висоту та сторону основи правильної трикутної призми. Обчислити її об'єм.
10. Дано сторони трикутника. Знайти його медіани.
11. Дано сторони трикутника. Знайти його бісектриси.

12. Дано радіус кола. Знайти площу квадрата, вписаного у коло.
13. Дано площу основи та висоту циліндра. Обчислити його об’єм.
14. Дано висоту та сторону основи правильної чотирикутної піраміди. Обчислити площу її бічної поверхні.
15. Дано радіус кулі. Обчислити об’єм кульового сегмента, якщо відома його висота.

Підвищений рівень

А

1. Визначити **h** – повну кількість годин та **m** – повну кількість хвилин від початку доби до того моменту (у першій половині доби), коли стрілка, що показує години, повернулася на **f** градусів (0<=**f**<360, **f** – ціле число).
2. Присвоїти цілій змінній **d** першу цифру із дробової частини додатного дійсного числа **x** (наприклад, якщо **x**=32.597, то **d**=5).
3. Нехай **k** – ціле число днів від 1 до 365. Присвоїти цілій змінній **n** значення 1,2,…,6 або 7 залежно від того, на який день тижня (понеділок, вівторок, …, суботу або неділю) припадає **k**-й день не високосного року, у якому 1 січня – це понеділок.
4. Цілій змінній **s** присвоїти суму цифр тризначного цілого числа **k.**
5. Дано ціле п’ятизначне число. Підрахувати суму цифр, які стоять на парних позиціях в числі, та добуток цифр, які стоять на непарних позиціях.
6. Два моменти часу однієї доби виражено годинами **h1**, **h2** та хвилинами **m1**, **m2** відповідно. Причому перший момент часу передує другому. Визначити інтервал часу у годинах та хвилинах між цими двома моментами.Триває **k**-та секунда доби. Визначити, скільки повних годин (**h**) та хвилин (**m**) пройшло до цього моменту (наприклад, якщо **k**=13257=3*3600+40*60+57, то **h**=3, **m**=40).
8. Паралелограм ABCD заданий координатами трьох його вершин:A(x_1,y_1), B(x_2,y_2), C(x_3,y_3). Знайти довжини сторін та площу паралелограма.
9. Вартість товару **n** гривень. Яку найменшу кількість банкнот по 20, 10 та 1 гривні треба витратити на купівлю товару?
10. Визначити **f** – кут (у градусах) між положенням годинної стрілки на початку доби та її положенням у **h** годин, **m** хвилин та **s** секунд (0<=**h**<=11, 0<=**m**,**s**<=59).
11. Пляшка води коштує 45 копійок. Порожні пляшки здають по 20 копійок, і на отримані гроші знову купують воду. Яку найбільшу кількість пляшок води можна купити за **S** копійок?
12. Визначити номери під’їзду та поверху за номером квартири дев’ятиповерхового будинку, якщо на кожному поверсі рівно 4 квартири, а нумерація квартир починається з першого під’їзду.
13. Визначити нормальну вагу людини та індекс маси її тіла за формулами: **h·t/240** та **m/h^2**, де **h** – зріст людини (у сантиметрах для першої формули та у метрах – для другої); **t** – довжина обхвату грудей (в см); **m** – вага (у кг). Індекс маси тіла прийнятий Всесвітньою організацією охорони здоров’я. Він не повинен перевищувати 25 пунктів.
14. Дано цілі числа **m**, **n** (0<**m**<=12, 0<**n**<=60), що вказують момент часу: “**m** годин, **n** хвилин”. Визначити найменший час (кількість повних хвилин), через який годинна та хвилинна стрілки на циферблаті співпадуть.
15. Дано коефіцієнти та вільні члени системи лінійних рівнянь

$$\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}.$$

Знайти розв’язок системи за умови, що визначник системи не дорівнює нулю.

Б

1. Дано число x>0. Обчислити $y = \dfrac{\sqrt[5]{\log_{3.2}^{2} x} + \sin x}{x = 2}$.
2. Дано число $x \neq 0$. Обчислити $y = \dfrac{\sqrt[3]{arcctg\, x} \cdot \lg\left|x^2 + x^3\right|}{\left|x\right| + 4}$.
3. Дано числа x, y. Обчислити $z = \dfrac{4^{x+y+8} \cdot \sin^2(x + y + 8)}{2 + \cos^3(x + y + 8)}$.
4. Дано число x, 0<=x<1. Обчислити $y = \dfrac{\arcsin x + \log_3(x^2 + 3x + 2)}{e^{x^2+3x+2}}$.
5. Дано число x. Обчислити $y = \dfrac{3 + \sin^2(8x^4 + 3)}{4 + \cos^2(8x^4 + 3)} + 8^{8x^4+3}$.
6. Дано число x, 0<=x<1. Обчислити $y = \dfrac{\arcsin x + \arccos x + arctg\, x}{\left|x^2 + 3x + 4\right|}$.
7. Дано додатні числа x, y. Обчислити $z = 3x^{4y} + e^{8x} + y^{\sin x} \cdot x^{\cos y}$.
8. Дано числа x, y. Обчислити $z = \dfrac{\sqrt[3]{\sin x} \cdot \sqrt[4]{e^{x+y}}}{2^{xy}}$.
9. Дано число x. Обчислити $y = \log_4(2^x + 3^x + 4^x) \cdot \sin^3(x^3 + 3)$.
10. Дано число x, 0<x<1. Обчислити $y = \dfrac{x + \arcsin x}{2 + \arccos x} + \sqrt[5]{arcctg\, x}$.
11. Дано число x. Обчислити $y = \dfrac{\sin^2(x^3 + 3x^2 + 4x + 5)}{2 + \cos^2(x^3 + 3x^2 + 4x + 8)} + \left|x^3 + 3x^2 + 4x + 10\right|$.
12. Дано додатні числа x, y. Обчислити $z = (x + 4)^{3y-2} + \lg(x^3 + y^3) - \sqrt[6]{x^3 + y^3}$.
13. Дано додатне число x. Обчислити $y = \dfrac{(8x + 4)^{2x+1} \cdot \sqrt[5]{3x^2 + 4x + 5}}{2^{x+4}}$.
14. Дано додатні числа x, y. Обчислити $z = \dfrac{\sqrt[5]{\sin^2(3x + 4y)}}{3^{3x+4y}} + \sqrt[6]{4^{3x+4y}}$.
15. Дано додатні числа x, y. Обчислити $z = 3^{x^y} \cdot \sqrt[4]{x^y} + \operatorname{tg}(x + y)$

Питання для самоконтролю

1. З яких символів складається алфавіт мови С?
2. Яка структура програми мови С?
3. Який елемент є обов'язковим для кожної С-програми?
4. Призначення оператора **return 0** в тілі функції main.
5. Призначення директиви **#include**.
6. Якими засобами здійснюється введення-виведення даних програми?
7. Поясніть призначення параметрів специфікації формату. Вкажіть можливі значення цих параметрів.

   %[прапорці][ширина][.точність][модифікатор]символ_специфікації
8. Наведіть приклади викликів функцій printf та scanf та поясніть їх призначення.

9. Які з наступних послідовностей символів є ідентифікаторами (іменами) змінних?
   а) x б) x1 с) x’ д) x1x2
   е) abcd ж) sin з) a-1 к) об’єм
10. Які типи даних мови С ви знаєте?
11. Наведіть приклади оголошень змінних.
12. Наведіть приклади оголошень змінних.
13. Наведіть приклади означень змінних.
14. Які з наступних послідовностей символів є присвоюваннями:
   а) a=b г) a*x+b=0 ж) z+=1,2
   б) a=c+1 д) z:=0 з) z++1
   в) a:b-sqr(2) е) y:=y к) –y=y
15. Відкомпілювати та запустити програму на виконання.
16. Записати на диск свою програму.
17. Зчитати свою програму з диску.

Практичне заняття № 2

Тема: Умовні оператори

Студент повинен знати: структуру програми, прості числові типи даних, синтаксис і семантику умовного оператора та оператора перемикача.

Теоретичні відомості

У мові С базова алгоритмічна структура розгалуження реалізується за допомогою умовного оператора IF. Його синтаксис такий:

if (<вираз>) <оператор1> **else** <оператор2> (1)

або **if** (<вираз>) <оператор> (2)

Оператор (1) реалізує повне розгалуження, а (2) – неповне. Слова **if**, **else** є службовими і перекладаються відповідно **якщо**, **інакше.** <вираз> - довільний вираз мови С. Кожний із операторів <оператор1>, <оператор2> чи <оператор> повинен сприйматися як один неділимий оператор. Тому використовують складений оператор у випадку, коли декілька операторів потрібно реалізувати у вигляді одного неділимого оператора. Внутрішні оператори умовного оператора **if** можуть бути будь-якими операторами мови С, зокрема умовними операторами. Це дозволяє вкладати один умовний оператор всередині другого.

Операційна семантика оператора (1) така:

1. Обчислюється <вираз>;
2. Якщо його значення не дорівнює 0, то виконується <оператор1>, інакше виконується <оператор2>;
3. Управління передається наступному оператору програми.

Операційна семантика оператора (2) така:

1. Обчислюється <вираз>;
2. Якщо його значення не дорівнює 0, то виконується <оператор>, інакше нічого не виконується;
3. Управління передається наступному оператору програми.

У мові С немає явного булевого типу зі значеннями true та false. Тому, значення <виразу> відмінне від нуля є аналогом true, а значення <виразу> рівне нулю є аналогом false.

Синтаксис оператора перемикача має вигляд:

```
switch (<вираз>)
{
    case <константа_1> : <оператори_1>;
    case <константа_2> : <оператори_2>;
    …….
    case <константа_k> : <оператори_k>;
    default : <оператори_k+1>;
}
```

або

```
switch (<вираз>)
{
    case <константа_1> : <оператори_1>;
    case <константа_2> : <оператори_2>;
    …….
    case <константа_k> : <оператори_k>;
}
```

Перший варіант оператора перемикача задає його повну форму, а другий – неповну. <вираз> - довільний вираз цілочислового типу або переліку. <константа_1>, <константа_2>, …, <константа_k> - це константи вибору, які є константами або

константними виразами, що мають цілочислове або символьне значення. Їхній тип повинен співпадати з типом <виразу>.

Операційна семантика повного варіанту оператора перемикача така:

1. Обчислюється значення <виразу>;
2. Це значення послідовно порівнюється зі значеннями констант вибору: <константа_1>, <константа_2>, …, <константа_k> - доки не буде знайдено відповідну константу;
3. Якщо значення <виразу> дорівнює знайденій константі, наприклад, <константі_n>, то виконуються <оператори_n> і всі наступні внутрішні оператори **switch**, тобто оператори <оператори_n+1>, …, <оператори_k>, <оператори_k+1>. Виконуємо перехід на пункт 5;
4. Якщо значення <виразу> не дорівнює жодній із констант вибору <константа_1>, <константа_2>, …, <константа_k>, то виконуються <оператори_k+1>, що розташовані в альтернативному варіанті **default**;
5. Управління передається наступному оператору програми.

Для переривання роботи оператора перемикача використовується оператор break.

Приклад 1

Дано три дійсні додатні числа. Визначити, чи можна побудувати трикутник з такими довжинами сторін.

Аналіз задачі

Трикутник можна побудувати за трьома відрізками a, b, c, якщо одночасно виконуються нерівності a+b>c, a+c>b, b+c>a – умови існування трикутника.

Алгоритм очевидний і наведений в тексті програми.

```
#include "stdafx.h"
#include "windows.h"
#include "locale.h"
int _tmain(int argc, _TCHAR* argv[])
{
	setlocale (0,"");
	double a, b, c;
	printf("Введіть три сторони трикутника\n");
	scanf("%lf%lf%lf", &a, &b, &c);
	if ((a<b+c) && (b<a+c) && (c<a+b))		//перевірка існування трикутника
		printf("Трикутник можна побудувати\n");
	else
		printf("Трикутник побудувати неможливо\n");
	system("pause");
	return 0;
}
```

```
Введіть три сторони трикутника
5,4  2,1  6,3
Трикутник можна побудувати
Для продолжения нажмите любую клавишу . . .
```

```
Введіть три сторони трикутника
10,43  2,87  6,34
Трикутник побудувати неможливо
Для продолжения нажмите любую клавишу . . .
```

Результати роботи програми.

Приклад 2

Обчислити значення функції:

$$y = \begin{cases} 5 * x, x = 31 \\ \cos|x|, x = 32 \\ tg^2(2x), x = 33 \\ 10, x = 34{,}35 \\ \cos(x) * \sin(x), \ в\,інших\,випадках. \end{cases}$$

Аналіз задачі

У задачі потрібно обчислити **y** для цілих значень **x**. Тому для вибору варіантів обчислення **y** потрібно використати оператор перемикач. Якби значення **x** були б дійсними, то для розв'язання задачі такого виду треба скористатися вкладеними операторами if.

```
#include "stdafx.h"
#include "windows.h"
#include "locale.h"
#include "math.h"

int _tmain(int argc, _TCHAR* argv[])
{
	setlocale (0,"");
	int x;
	double y;
	printf("Введіть значення x = ");
	scanf("%lf", &x);
	switch (x)
		{	case 31: y = 5*x; break;
			case 32: y = cos((double)abs(x)); break;
			case 33: y = tan(2.*x)*tan(2.*x); break;
			case 34:
			case 35: y = 10; break;
			default: y = cos((double)x)*sin((double)x);
		}
	printf("Значення y = %lf\n", y);
	system("pause");
	return 0;
}
```

```
Введіть значення x = 31
Значення y = 155,000000
Для продолжения нажмите любую клавишу . . .
```

Результати роботи програми.

Задачі

1. Дано число x. Обчислити y(x):

$$y(x) = \begin{cases} 0, \text{ якщо } x \le 0; \\ x^2 - x, \text{ якщо } 0 < x \le 1; \\ x^2 - \sin \pi x^2 - 1, \text{ в іншому випадку.} \end{cases}$$

2. Написати програму обчислення коренів квадратного рівняння $ax^2+bx+c=0$.

3. Дано числа x, y, z. Обчислити $V=\min(x^2+y^2+z^2, (x+y+z)^2)-3$.
4. Дані числа a_1, b_1, c_1, a_2, b_2, c_2. Знайти координати перетину прямих, що задаються рівняннями $a_1x+b_1y=c_1$ і $a_2x+b_2y=c_2$ або повідомити, що прямі співпадають, паралельні, не існують.
5. Дано розміри a, b, c цеглини і розміри x, y прямокутного отвору. Вияснити, чи пройде цеглина в отвір. Просувати цеглину в отвір дозволяється тільки так, щоб кожне її ребро було паралельне або перпендикулярне кожній із сторін отвору.

☺ Індивідуальні завдання

Основний рівень

А

1. Визначити, чи є серед трьох цілих чисел **a**, **в**, **c** хоча б одна пара рівних між собою.
2. Дано прямокутник зі сторонами **a**, **b** і квадрат із стороною **c**. З'ясувати, чи поміститься прямокутник у квадрат.
3. Поміняти місцями три різних дійсних числа **x, y, z** так, щоб вони утворили зростаючу послідовність.
4. Визначити, чи є серед цифр заданого натурального тризначного числа однакові.
5. Визначити, чи дорівнює квадрат заданого двозначного числа кубу суми цифр цього числа.
6. Дано тризначне натуральне число. Скільки в ньому різних цифр?
7. Визначити, чи дорівнює сума двох перших цифр заданого натурального чотиризначного числа сумі двох його останніх цифр.
8. Знайти найменше із трьох різних даних дійсних чисел.
9. Дано один прямокутник зі сторонами **a**, **b** і другий прямокутник зі сторонами **c**, **d**. З'ясувати, чи можна перший прямокутник розмістити всередині другого так, щоб сторони прямокутників були паралельні.
10. Дано два циліндри з радіусами основ R1, R2 і висотами H1, H2 відповідно. З'ясувати, чи можна розмістити один циліндр всередині другого так, щоб їхні твірні були паралельними.
11. Визначити, чи є даний рік високосним (рік з двома нулями в кінці високосний, коли число ділиться на 400).
12. Дано три різних дійсних числа. Знайти те з них, яке за величиною міститься між двома іншими.
13. Надрукувати фразу "мені **n** рік (роки, років)" так, щоб відмінок слова "рік" узгоджувався з числом **n**, яке вводиться з клавіатури.
14. Поміняти місцями три різних дійсних числа **x, y, z** так, щоб вони утворили спадну послідовність.
15. Знайти найбільше із трьох різних даних дійсних чисел.

Б

Підрахувати та вивести на екран значення функції y(x), де x – ціле число:

1. $y=\begin{cases} 0, x=1 \\ x^2+2, x=3 \\ |x|+\cos|x|, x=5 \\ |tg(x)+x^3|, x=7{,}8 \\ x, \text{в інших випадках} \end{cases}$

2. $y=\begin{cases} 10, x=20 \\ \ln|x|, x=10 \\ |x^4|, x=1 \\ tg(x)+|x|, x=5{,}6 \\ arctg(x), \text{в інших випадках} \end{cases}$

3. $y=\begin{cases} 6, x=-1 \\ |tg(x)|, x=0 \\ \sin(x)+|x|, x=2 \\ arctg(x)+|x|, x=5,6 \\ 10, \text{в інших випадках} \end{cases}$

4. $y=\begin{cases} |\cos(x)|, x=1 \\ \sin(x), x=2 \\ tg(x), x=5 \\ x+|x^3|, x=9,10 \\ x^2+2x+3, \text{в інших випадках} \end{cases}$

5. $y=\begin{cases} |x^3|, x=5 \\ 3, x=3 \\ |3*tg|x||, x=6 \\ \cos^2(x), x=10,11 \\ \cos(x)*\sin(x), \text{в інших випадках} \end{cases}$

6. $y=\begin{cases} tg^2|x|, x=11 \\ \cos(x), x=12 \\ 3x^2+x, x=13 \\ \sin^2(x), x=14,15 \\ 5, \text{в інших випадках} \end{cases}$

7. $y=\begin{cases} |x|, x=20 \\ \cos(x), x=21 \\ \sin^2|x|, x=22 \\ arctg(x^2), x=23,24 \\ 24, \text{в інших випадках} \end{cases}$

8. $y=\begin{cases} |x^2+5|, x=2 \\ arctg(x), x=7 \\ \cos^2|x|, x=-2,-3,-4 \\ x+x^2+x^3, x=8,...,16 \\ 66, \text{в інших випадках} \end{cases}$

9. $y=\begin{cases} 10, x=1 \\ -10, x=-1 \\ \lg|x|, x=11,...,20 \\ tg(x)+|x|, x=5,6 \\ tg(x), \text{в інших випадках} \end{cases}$

10. $y=\begin{cases} |x^2+5x+-6|, x=1,...,7 \\ \sin(x), x=-2 \\ ctg(x), x=15 \\ \ln(x)+|x^3|, x=9,10 \\ 33, \text{в інших випадках} \end{cases}$

11. $y=\begin{cases} x+2\ln|x|, x=-1,...,1 \\ \{\cos(x)\}, x=5,...,10 \\ x^2+3, x=3 \\ |x|+\cos|x|, x=4 \\ x, \text{в інших випадках} \end{cases}$

12. $y=\begin{cases} \sqrt{x-3}, x=4,..10 \\ \cos(x), x=-10,..0 \\ \sin(x)+\sin^2(2x), x=1,2,3 \\ 22, \text{в інших випадках} \end{cases}$

13. $y=\begin{cases} \sqrt[3]{x^3+5}, x=1,..,3 \\ \lg(x)+\lg(x+5), x=5 \\ \cos(x)+\sin^2(2x), x=1,2,3 \\ 12, \text{в інших випадках} \end{cases}$

14. $y=\begin{cases} 2x+5, x=5 \\ \sin(x), x=-10,..0 \\ \sqrt{x-1}+\sqrt{x-2}, x=3,4 \\ -2, \text{в інших випадках} \end{cases}$

15. $y=\begin{cases} tg(x)+3, x=-3,..-1 \\ 2\cos(2x), x=2 \\ x^x, x=3,4 \\ 1001, \text{в інших випадках} \end{cases}$

Підвищений рівень

А

Дано число х. Обчислити у(х):

1. $y=\begin{cases}\dfrac{1}{x^2},\ якщо\ x\le -1;\\ -x,\ якщо\ -1<x\le 0;\\ x^2,\ якщо\ 0<x<1;\\ 1,\ в\ іншому\ випадку.\end{cases}$

2. $y=\begin{cases}x+2,\ якщо\ x\le -2;\\ 0,\ якщо\ -2<x\le 0;\\ \dfrac{x}{2},\ якщо\ 0<x\le 2;\\ \sqrt{x-1},\ в\ іншому\ випадку.\end{cases}$

3. $y=\begin{cases}(x+1)^2,\ якщо\ x\le -1;\\ x+1,\ якщо\ -1<x\le 0;\\ 2^x,\ якщо\ 0<x\le 2;\\ 4,\ в\ іншому\ випадку.\end{cases}$

4. $y=\begin{cases}\cos 3x,\ якщо\ x\le 0;\\ -x+1,\ якщо\ 0<x\le 1;\\ (x-1)^2,\ якщо\ 1<x\le 3;\\ 4,\ в\ іншому\ випадку.\end{cases}$

5. $y=\begin{cases}3x+15,\ якщо\ x\le -4;\\ 3,\ якщо\ -4<x\le -2;\\ x^2-1,\ якщо\ -2<x\le 1;\\ \log_{1/2}x,\ в\ іншому\ випадку.\end{cases}$

6. $y=\begin{cases}\sin\dfrac{\pi x}{4},\ якщо\ x\le 0;\\ x^2,\ якщо\ 0<x\le 2;\\ 3x-2,\ якщо\ 2<x\le 3;\\ \sqrt{4x^2+2x+5},\ в\ іншому\ випадку.\end{cases}$

7. $y=\begin{cases}4^{x+2},\ якщо\ x\le -2;\\ -2x-3,\ якщо\ -2<x\le 0;\\ -3,\ якщо\ 0<x\le 8;\\ \log_{1/2}x,\ в\ іншому\ випадку.\end{cases}$

8. $y=\begin{cases}\sqrt[3]{x+1},\ якщо\ x\le -2;\\ -|x|+1,\ якщо\ -2<x\le 0;\\ \arcsin x,\ якщо\ 0<x\le 1;\\ \pi/2,\ в\ іншому\ випадку.\end{cases}$

9. $y=\begin{cases}x^{2/3}+x^{1/3},\ якщо\ x\le 0;\\ \sqrt{x},\ якщо\ 0<x\le 4;\\ 2,\ якщо\ 4<x\le 9;\\ \log_3 x,\ в\ іншому\ випадку.\end{cases}$

10. $y=\begin{cases}\pi,\ якщо\ x\le -1;\\ \arccos x,\ якщо\ -1<x\le 1;\\ x^2-1,\ якщо\ 1<x\le 3;\\ 4\sqrt{x+1},\ в\ іншому\ випадку.\end{cases}$

11. $y=\begin{cases}2^{-x},\ якщо\ x\le -1;\\ -2x,\ якщо\ -1<x\le 0;\\ tg\dfrac{\pi x}{4},\ якщо\ 0<x\le 1;\\ \sqrt{x},\ в\ іншому\ випадку.\end{cases}$

12. $y=\begin{cases}2/x,\ якщо\ x\le -2;\\ \sin\dfrac{\pi x}{4},\ якщо\ -2<x\le 0;\\ x^3,\ якщо\ 0<x\le 1;\\ x^{-2},\ в\ іншому\ випадку.\end{cases}$

13. $y=\begin{cases}arcctg\,x,\ якщо\ x\le 0;\\ \arccos x,\ якщо\ 0<x\le 1;\\ x+1,\ якщо\ 1<x\le 2;\\ \sqrt{2x+5},\ в\ іншому\ випадку.\end{cases}$

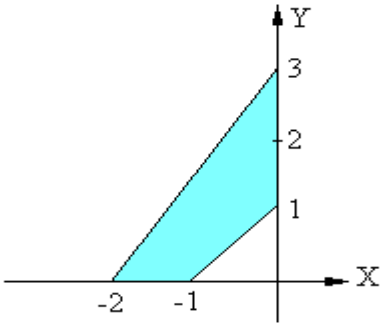
14. $y=\begin{cases}3^{x+1},\ якщо\ x\le -1;\\ |x|,\ якщо\ -1<x\le 0;\\ 0,\ якщо\ 0<x\le 1;\\ \log_{1/3}x,\ в\ іншому\ випадку.\end{cases}$

15. $y=\begin{cases}arctg\,x,\ якщо\ x\le 0;\\ \sqrt[3]{x},\ якщо\ 0<x\le 1;\\ \sqrt{x},\ якщо\ 1<x\le 4;\\ x-2,\ в\ іншому\ випадку.\end{cases}$

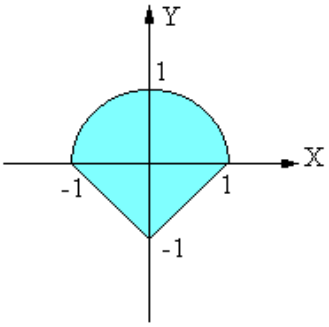
Б

Дані числа x, y. З’ясувати, чи належить точка (x, y) області, зображеній на малюнку.

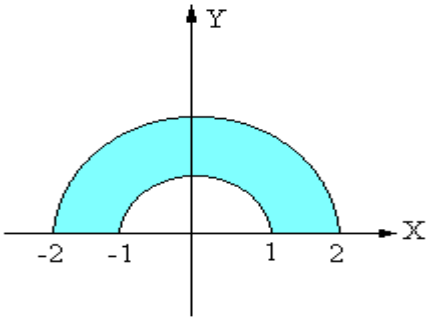
1.



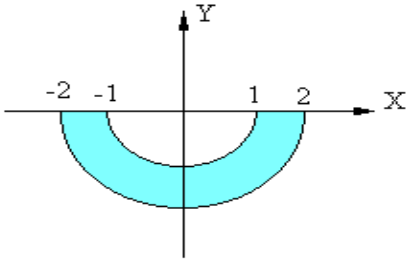
2.



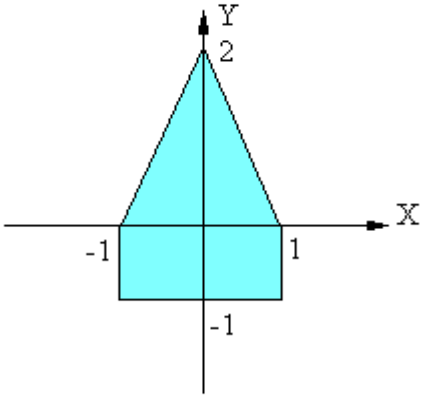
3.



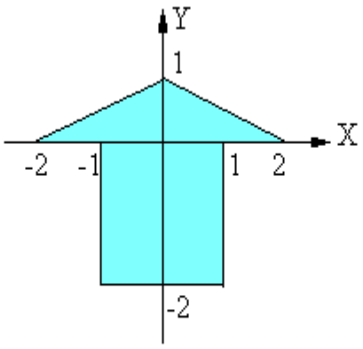
4.



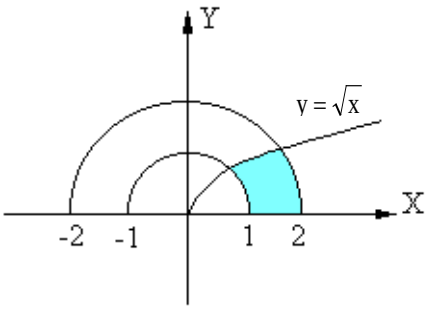
5.



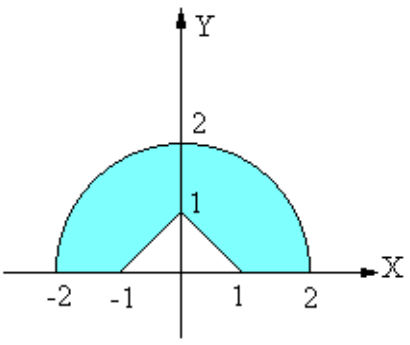
6.



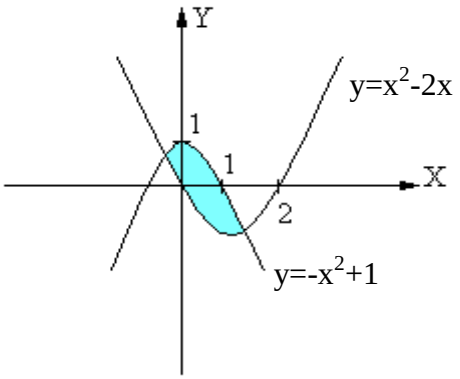
7.



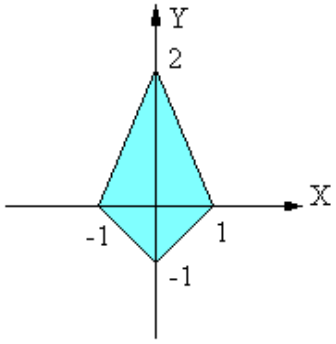
8.



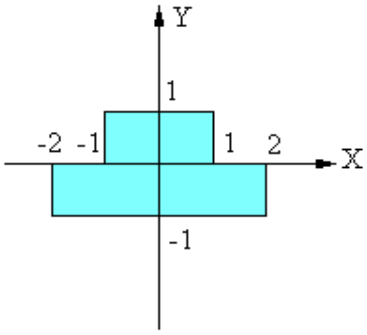
9.



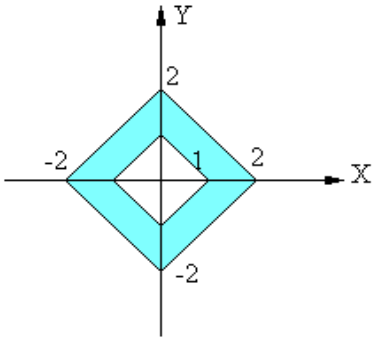
10.



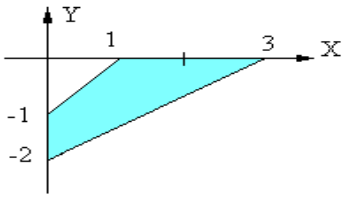
11.



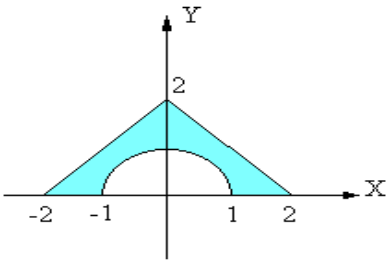
12.



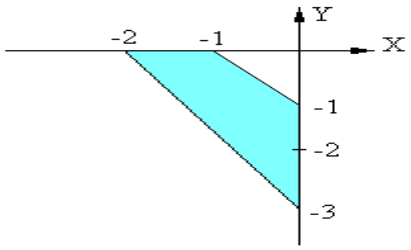
13.



14.



15.



☝ Питання для самоконтролю

1. Для чого і коли використовується умовний оператор? Чому він так називається?
2. Синтаксис умовного оператора.
3. Яким чином працює (яка семантика) умовний оператор?
4. Чим відрізняється повна і скорочена форми умовного оператора?
5. Коли використовується оператор перемикач?
6. Опишіть синтаксис оператора перемикача.
7. Опишіть роботу (семантику) оператора перемикача.
8. Якого типу може бути вираз в операторі перемикача?
9. Коли оператори беруть в операторні дужки?

Практичне заняття № 3

*Тема: **Цикли***

Студент повинен знати: структуру програми, прості числові типи даних, синтаксис і семантику операторів циклів.

Теоретичні відомості

Базові алгоритмічні конструкції повторення реалізуються у мові С через оператори циклів. Мова С містить три види циклів – **ДЛЯ**, **ПОКИ**, **ДО**.

Цикл **ДЛЯ** використовується найчастіше і реалізується оператором циклу з параметром (лічильником). Його синтаксис

for (<вираз_1>; <вираз_2> ; <вираз_3>) <оператор>;

<вираз_1> - встановлює початкові значення змінних циклу (параметрів), його називають виразом ініціалізації; <вираз_2> - задає умову виконання циклу, його називають виразом умови; <виразу_3> - виконує зміну значень змінних циклу, його називають виразом ітерації (приросту). <оператор> - довільний оператор, який називається тілом циклу.

Оператор **for** працює наступним чином:

1. Обчислюється <вираз_1>. Під час його обчислення параметри (змінні) циклу отримують початкові значення.
2. Обчислюється <вираз_2>, тобто перевіряється умова виконання циклу.
3. Якщо значення <виразу_2> дорівнює нулю (умова хибна), то виконання циклу завершується і керування передається оператору, наступному за **for.**
4. Якщо значення <виразу_2> не дорівнює нулю (умова істинна), то виконується <оператор> - тіло циклу.
5. Обчислюється <вираз_3>. Під час його обчислення параметри змінюють значення – отримують нові значення.
6. Відбувається перехід на крок 2.

Цикл **ПОКИ** (з передумовою) реалізується оператором **while**, який має наступний синтаксис

while (<вираз>) <оператор>.

Тут <вираз> - довільний вираз, який задає умову виконання циклу; <оператор> - довільний оператор, який називається тілом циклу.

Оператор **while** працює наступним чином:

1. Обчислюється <вираз>, тобто перевіряється умова виконання циклу.
2. Якщо значення <виразу> дорівнює нулю (умова хибна), то виконання циклу завершується і керування передається оператору, наступному за **while.**
3. Якщо значення <виразу> не дорівнює нулю (умова істинна), то виконується <оператор> - тіло циклу.
4. Відбувається перехід на крок 1.

Цикл **ДО** (з після умовою) реалізується оператором **do-while**, який має наступний синтаксис

do <оператор> **while** <вираз>

Тут <вираз> - довільний вираз, який задає умову виконання циклу; <оператор> - довільний оператор, який називається тілом циклу.

Оператор **do-while** працює наступним чином:

1. Виконується <оператор> - тіло циклу.
2. Обчислюється <вираз>, тобто перевіряється умова виконання циклу.
3. Якщо значення <виразу> дорівнює нулю (умова хибна), то виконання циклу завершується і керування передається оператору, наступному за **do-while.**
4. Якщо значення <виразу> не дорівнює нулю (умова істинна), то виконується перехід на крок 1.

Приклад 1

Підрахувати нескінченну суму із заданою точністю ε (ε>0). Вважати, що потрібна точність досягнута, якщо обчислена вже сума деяких перших доданків і наступний доданок став меншим за ε. Цей та всю решту доданків можна не враховувати.

$$\sum_{i=0}^{\infty} \frac{1}{4^i + 5^{i+2}}.$$

Аналіз задачі

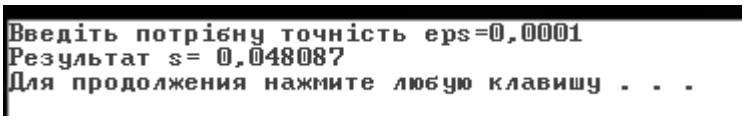
Позначимо значення шуканої суми через S, а суму перших k доданків через S_k. Якщо обчислена сума S_k, то наступна сума $S_{k+1}= S_k+a_{k+1}$, де a_{k+1} – (k+1)-й доданок. Остання рівність підказує ідею алгоритму розв'язку задачі. Початкове значення суми S треба покласти рівними 0. Потім послідовно виконувати обчислення за формулами $S_0=S+a_0$, $S_1=S_0+a_1$, $S_2=S_1+a_2$ і т. д. Такі обчислення легко реалізуються у програмі за допомогою циклічного виконання присвоювання S = S+a, де a – черговий доданок. При обчисленні сум S_k треба на кожному кроці визначати доданок a_k та порівнювати його з ε. Для багатьох випадків доданок a_k обчислюється рекурентно через попередній доданок. У нашій задачі використовувати рекурентну формулу недоцільно. Будемо послідовно обчислювати степені 4^i та 5^i, користуючись формулами $4^i=4^{i-1}*4$, $5^i=5^{i-1}*5$.

Алгоритм

1. Ввести точність - ε.
2. Присвоїти a=1, b=25, add=1/26 (змінна a – це 4^0, b - 5^2, add – a_0).
3. Присвоїти сумі s початкове значення 0 (s=0).
4. Повторювати в циклі ДО наступні дії
    a. Обчислити чергову суму $S_k=S_{k-1}+a_k$, виконавши присвоєння s=s+and або s+=and.
    b. Обчислити 4^{k+1}, виконавши присвоєння a=a*4 або a*=4.
    c. Обчислити 5^{k+3}, виконавши присвоєння b=b*5 або b*=5.
    d. Обчислити a_{k+1}, виконавши присвоєння add=1/(a+b).

    поки умова $a_{k+1}\geq\varepsilon$ істинна.
5. Вивести суму s.

```
#include "stdafx.h"
#include "windows.h"
#include "locale.h"
int _tmain(int argc, _TCHAR* argv[])
{
        setlocale (0,"");
        double a,b,s,add,eps;                   // Оголошення змінних
        printf ("Введіть потрібну точність eps=");
        scanf ("%lf",&eps);
        a=1;   b=25;  add=1./26;        s=0;    // Початкові значення змінних
        do                                      // Цикл ДО
        {
                s+=add;    a*=4;      b*=5;    add=1/(a+b);
        }
        while (add>=eps);
        printf ("Результат s= %lf\n",s);
        system ("pause");
        return 0;
}
```



Результат роботи програми.

Приклад 2

Користуючись розкладом у ряд для ln2

$$\ln 2 = \frac{2}{3}\cdot\left(1+\frac{1}{3}\cdot\frac{1}{9}+\frac{1}{5}\cdot\frac{1}{9^2}+\frac{1}{7}\cdot\frac{1}{9^3}+...\right),$$

обчислити ln2 з точністю до заданого ε>0. Необхідна точність вважається досягнутою, якщо черговий доданок менший ε.

Аналіз задачі

Основою алгоритму є цикл, у якому потрібно виконувати дві дії: обчислити черговий доданок ряду, додати його до знайденої суми.

Ряд $1+\frac{1}{3}\cdot\frac{1}{9}+\frac{1}{5}\cdot\frac{1}{9^2}+\frac{1}{7}\cdot\frac{1}{9^3}+...$ можна представити як нескінченну суму $a_0+a_1+\ldots+a_i+\ldots$ Тоді $a_i=\frac{1}{2i+1}\cdot\frac{1}{9^i}$, де i=0,1,2,… Член a_i містить степінь 9^i, який дуже швидко зростає. Часто 9^i обчислюють як pow(9, i). Однак функція pow повертає дійсний результат, який при великих i буде значно відрізнятися від справжнього значення 9^i. Зазвичай, коли члени ряду містять степені або факторіали створюють рекурентне співвідношення для обчислення наступного члена ряду через попередній. Це рекурентне співвідношення створюють, порівнюючи a_i з a_{i+1}.

Створимо це співвідношення. Маємо $a_{i+1}=\frac{1}{2(i+1)+1}\cdot\frac{1}{9^{i+1}}=\frac{1}{2i+3}\frac{1}{9^{i+1}}$. Тоді обчислюємо $a_{i+1}/a_i=\frac{(2i+1)\cdot 9^i}{(2i+3)\cdot 9^{i+1}}=\frac{2i+1}{(2i+3)\cdot 9}$. Отримуємо рівність $a_{i+1}=a_i\cdot\frac{2i+1}{(2i+3)\cdot 9}$. Це і є шукане рекурентне співвідношення, яке дозволяє обчислити (i+1)-й доданок через i-й. Це співвідношення виконується при i=0,1,…, причому a_0=1.

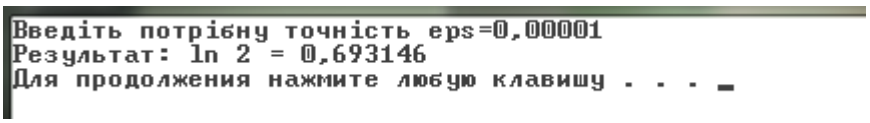
У програмі будемо використовувати цикл

while (a_i>=ε) {додати a_i до суми; i++; обчислити a_i;}.

Він припинить свою роботу, коли a_i стане меншим за ε. Тоді буде обчислено наближене значення ряду.

```
#include "stdafx.h"
#include "windows.h"
#include "locale.h"

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (0,"");
        double a, suma, eps;
        int i;
        printf ("Введіть потрібну точність eps=");
        scanf ("%lf",&eps);
        suma=1;                 // Початкове значення суми ряду, яке містить a0
        i=0;                    // Початкове значення індексу i
        a=1./27;                // Член ряду a1
        while (a>=eps)
        {       suma+=a;        // Добавлення ai до суми
```

```
        i++;                                    // Збільшення і
        a*=(double)(2*i+1)/((2*i+3)*9);         // Обчислення наступного a_{i+1}
    }
    printf ("Результат: ln 2 = %lf\n", 2./3*suma);
    system ("pause");
    return 0;
}
```



Результат роботи програми.

Задачі

1. Для даних **x** і **n** (**n** – натуральне) обчислити:
   a) $y = \sin x \cdot \sin x^2 \cdot \sin x^3 \cdot \ldots \cdot \sin x^n$;
   b) $y = \sin x + \sin x^2 + \sin x^3 + \ldots + \sin x^n$.
2. Стандартні функції представлені рядами:
   a) $y = e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots \qquad (0 \le x < 1)$,
   b) $y = \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots + \frac{(-1)^{n-1} x^n}{n!} + \ldots \qquad (|x| < 1)$,
   c) $y = \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} + \ldots \qquad \left(|x| < \frac{\pi}{4}\right)$,
   d) $y = \cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + \frac{(-1)^n x^{2n}}{(2n)!} + \ldots \qquad \left(|x| < \frac{\pi}{4}\right)$,
   e) $y = \operatorname{arctg} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \ldots + \frac{(-1)^n x^{2n+1}}{2n+1} + \ldots \qquad (|x| < 1)$.

   Задавши x, обчислити значення функції з точністю до ε>0. Необхідна точність вважається досягнутою, якщо черговий доданок за модулем менший ε. Усі наступні доданки можна не враховувати.
3. З точністю до ε>0 знайти корінь рівняння методом поділу відрізка навпіл:
   a) $\operatorname{arctg} x - \ln x = 0$;
   b) $x^3 + x^2 + x + 1 = 0$ на відрізку [-2; 1];
   c) $x^2 \cos 2x + 1 = 0$ на відрізку $[0; \pi/2]$;
   d) $2 \operatorname{arctg} x - e^{2x} = 0$.
4. Знайти канонічний розклад даного натурального числа.
5. Результат лижника в перегонах задається трійкою: його стартовим номером, кількістю хвилин і секунд. Вводяться послідовно результати перегонів **n** лижників (**n**>3). Написати програму, яка після введення чергового результату виводить завжди поточну трійку найкращих результатів.
6. Дано натуральне число k. Знайти k-ту цифру послідовності:
   a) 149162536…, у якій виписані підряд квадрати всіх натуральних чисел;
   b) 1123581321…, у якій виписані підряд всі числа Фібоначі.

☺ Індивідуальні завдання

Основний рівень

А

Дано натуральне число **n** та дійне число **x**. Обчислити:

1. $\frac{x^1}{2!}+\frac{x^2}{3!}+\ldots+\frac{x^n}{(n+1)!}$;
2. $\left(\frac{1}{1!}+\sqrt{|x|}\right)+\left(\frac{1}{2!}+\sqrt{|x|}\right)+\ldots+\left(\frac{1}{n!}+\sqrt{|x|}\right)$;
3. $\left(1+\frac{\sin x}{1!}\right)\cdot\left(1+\frac{\sin 2x}{2!}\right)\cdot\ldots\cdot\left(1+\frac{\sin nx}{n!}\right)$;
4. $\left(\frac{1}{2}-\cos|x|\right)\cdot\left(\frac{2}{3}-\cos^2|x|\right)\cdot\ldots\cdot\left(\frac{n}{n+1}-\cos^n|x|\right)$.

Дано натуральне число **n**. Обчислити:

5. $\frac{1}{2^2}+\frac{1}{3^2}+\ldots+\frac{1}{(n+1)^2}$;
6. $\frac{1}{1^3}+\frac{1}{2^3}+\ldots+\frac{1}{n^3}$;
7. $\frac{1}{2!}-\frac{2}{3!}+\ldots+\frac{(-1)^{n+1}\cdot n}{(n+1)!}$;
8. $\frac{1}{2^2}+\frac{1}{4^2}+\frac{1}{8^2}+\ldots+\frac{1}{2^{2n}}$;
9. $\frac{1^2}{1^2+5}\cdot\frac{2^2}{2^2+5}\cdot\ldots\cdot\frac{n^2}{n^2+5}$;
10. $\frac{1}{1^4}+\frac{1}{2^4}+\ldots+\frac{1}{n^4}$;
11. $\frac{1}{3^2}+\frac{1}{5^2}+\ldots+\frac{1}{(2n+1)^2}$;
12. $-\frac{1}{2}+\frac{1}{4}-\ldots+\frac{(-1)^n}{2n}$;
13. $\frac{1}{1\cdot 2}-\frac{1}{2\cdot 3}+\ldots+\frac{(-1)^{n+1}}{n\cdot(n+1)}$;
14. $-\frac{2}{1!}+\frac{3}{2!}-\ldots+\frac{(-1)^n(n+1)}{n!}$;
15. $\frac{1}{2^2}+\frac{1}{4^2}+\ldots+\frac{1}{(2n)^2}$;
16. $\frac{1}{1^5}+\frac{1}{2^5}+\ldots+\frac{1}{n^5}$.

Б

Обчислити наближене значення нескінченної суми з точністю до заданого ε>0. Наближене значення вважати отриманим, якщо вже обчислена сума декількох перших доданків, а черговий доданок виявився за модулем меншим ε. Цей доданок теж треба потім додати до суми.

1. $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{(-1)^n}{n}+\ldots$;
2. $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots+\frac{(-1)^n}{2n+1}+\ldots$;
3. $\frac{1}{1\cdot 2!}+\frac{1}{2\cdot 3!}+\frac{1}{3\cdot 4!}+\ldots+\frac{1}{n\cdot(n+1)!}+\ldots$;
4. $1+\frac{2}{2}+\frac{3}{2^2}+\frac{4}{2^3}+\ldots+\frac{n}{2^{n-1}}+\ldots$;
5. $1+\frac{2}{1!}+\frac{2^2}{2!}+\frac{2^4}{3!}+\ldots+\frac{2^n}{n!}+\ldots$;
6. $1-\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}-\frac{1}{3\cdot 4}+\ldots+\frac{(-1)^n}{n\cdot(n+1)}+\ldots$.

Функції подано у вигляді рядів. Задавши x, обчислити значення функції з точністю до заданого ε>0. Необхідна точність вважається досягнутою, якщо черговий доданок за модулем менший ε. Усі наступні доданки можна не враховувати.

7. $y=\operatorname{sh}x=x+\frac{x^3}{3!}+\frac{x^5}{5!}+\ldots+\frac{x^{2n-1}}{(2n-1)!}+\ldots$, $\left(|x|<\frac{\pi}{4}\right)$;

8. $y = \operatorname{ch} x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + ... + \frac{x^{2n}}{(2n)!} + ...$, $\left(|x| < \frac{\pi}{4}\right)$;

9. $y = \ln\frac{1+x}{1-x} = 2x\left(1 + \frac{x^2}{3} + \frac{x^4}{5} + ... + \frac{x^{2n}}{2n+1} + ...\right)$, $(|x| < 1)$;

10. $y = \frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + ... + nx^{n-1} + ...$, $(|x| < 1)$;

11. $y = \frac{1}{e^x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - ... + \frac{(-1)^n x^n}{n!} + ...$, $(|x| < 1)$.

Біноміальний ряд $(1+x)^n = 1 + n \cdot x + \frac{n(n-1)}{2!} \cdot x^2 + \frac{n(n-1)(n-2)}{3!} \cdot x^3 + ...$ використовується для наближених обчислень коренів. З точністю до ε>0 обчислити наступні корені через біноміальний ряд. Необхідна точність вважається досягнутою, якщо черговий доданок за модулем менший ε. Усі наступні доданки можна не враховувати.

12. $\sqrt{2} = 1.4 \cdot \left(1 + \frac{1}{49}\right)^{\frac{1}{2}}$;

13. $\sqrt{2} = 1.4 \cdot \left(1 - \frac{1}{50}\right)^{-\frac{1}{2}}$;

14. $\sqrt[3]{2} = \frac{5}{4} \cdot \left(1 + \frac{3}{125}\right)^{\frac{1}{3}}$;

15. $\sqrt[3]{3} = \frac{10}{7} \cdot \left(1 + \frac{29}{1000}\right)^{\frac{1}{3}}$.

16. Число π може бути представлене у вигляді суми двох рядів

$$\frac{\pi}{4} = \left(\frac{1}{2} - \frac{1}{3} \cdot \frac{1}{2^3} + \frac{1}{5} \cdot \frac{1}{2^5} - ...\right) + \left(\frac{1}{3} - \frac{1}{3} \cdot \frac{1}{3^3} + \frac{1}{5} \cdot \frac{1}{3^5} - ...\right).$$

Обчислити число π із заданою точністю ε>0. Необхідна точність вважається досягнутою, якщо черговий доданок обох рядів менший ε.

В

1. Знайти перший від'ємний член послідовності cos(ctg n), n=1,2,3, …
2. Дано натуральне число n. Обчислити $\sqrt{3 + \sqrt{6 + \mathrm{L} + \sqrt{3(n-1) + \sqrt{3n}}}}$.
3. Дано натуральне число n. Обчислити $p = (1 - \frac{1}{2^2})(1 - \frac{1}{3^2})\mathrm{L}\ (1 - \frac{1}{n^2})$, n>2.
4. Обчислити y=cos(1+cos(2+…+cos(39+cos40) …)).
5. Дано непарне натуральне n. Обчислити n!!=1*3*5*…*n.
6. Дано парне натуральне n. Обчислити n!!=2*4*…n.
7. Дано натуральне число. Знайти добуток його цифр.
8. Дано натуральне число. Знайти число, яке отримується записуванням цифр заданого числа у зворотному порядку.
9. Дано деяке натуральне число. Знайти в ньому цифру, що стоїть на к-й позиції.
10. Дано 10 цілих чисел. Визначити, скільки з них приймають найбільше значення.
11. Дані цілі $a_1,...a_{10}$. Отримати суму тих членів a_i даної послідовності, які є непарними і від'ємними.
12. Дано натуральне число. Знайти суму його цифр.
13. Дано 10 цілих чисел. Визначити, скільки з них приймають найменше значення.
14. Дано 10 цілих чисел. Визначити, скільки разів серед них зустрічається число 0.
15. Дано 10 цілих чисел. Визначити, скільки з них від'ємних чисел.

Підвищений рівень

А

1. Дано n дійсних чисел. Знайти порядковий номер того з них, яке найближче до цілого.
2. З'ясувати, чи є задане натуральне число паліндромом, тобто таким, у якого десятковий запис читається однаково зліва направо і справа наліво.
3. Обчислити кількість точок із цілочисловими координатами, що попали в круг із центром у початку координат та радіусом R (R>0).
4. Знайти всі тризначні натуральні числа рівні сумі факторіалів своїх цифр.
5. Вивести всі натуральні числа менші за n, сума цифр яких рівна m.
6. Підрахувати суму квадратів усіх цілих чисел, що попадають в інтервал ($\ln x$, e^x), x>1.
7. Дані натуральні числа m, n (m<n). Записати дріб m/n у вигляді нескоротного дробу.
8. Дано дійсне число x. Послідовність $a_1,a_2,\ldots$ створено за таким законом:

$$a_n = \frac{(-1)^n x^{2n}}{n(n+1)(n+2)}$$

Одержати $a_1+a_2+\ldots+a_к$, де к – мінімальне ціле число, що задовольняє умови: k>10, $|a_{k+1}|<10^{-5}$.

9. Знайти всі натуральні числа менші за 100, які є паліндромами та при піднесенні до квадрата також утворюють паліндроми.
10. Обчислити дріб $\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cfrac{1}{\ddots\,\cfrac{1}{101+\cfrac{1}{103}}}}}}$
11. Дано натуральне число n, дійсні числа x, a_0, a_1, ..., a_n. Обчислити за схемою Горнера значення многочлена $a_0x^n+a_1x^{n-1}+...+a_n$ у точці x.
12. Гусінь повзе по гумці зі швидкістю 1 см/хв, прагнучи досягти протилежного кінця. Гумка має довжину 7см і може розтягуватися до будь-якої довжини. Через кожну хвилину гумку розтягують на 7см (через 1хв. довжина гумки подвоюється – 14см, через 2хв. потроюється - 21см і т.д.). Гусінь міцно тримається на гумці та продовжує шлях. Чи дістанеться гусінь коли-небудь протилежного кінця? Якщо так, то коли?
13. Дана послідовність із n цілих чисел, серед яких є хоч одне від'ємне. Знайти величину найбільшого від'ємного числа цієї послідовності.
14. Дано натуральне число. Перевірити, чи є воно простим.
15. Дано натуральне число. Перевірити, чи є воно досконалим, тобто рівним сумі всіх своїх дільників, крім самого цього числа (наприклад, число 6 досконале, бо 6=1+2+3).
16. Дана послідовність із n дійсних чисел. Знайти кількість членів у найдовшій зростаючій підпослідовності даної послідовності.
17. Дана послідовність із n дійсних чисел. Знайти кількість членів у найдовшій підпослідовності із від'ємних чисел даної послідовності.
18. Дана послідовність натуральних чисел, яка закінчується нулем. Знайти три найбільших числа цієї послідовності.

Б

Функція означена та обмежена на відрізку [a,b]. Написати програму обчислення площі криволінійної трапеції, обмеженої цим відрізком, графіком функції та вертикальними прямими x=a і x=b, застосувавши формули прямокутників і трапецій. Порівняти результати, отримані за різними формулами. Формула прямокутників для наближеного обчислення площі має вигляд

$$\int_a^b f(x)dx \approx h\cdot\left(f\left(a+\frac{h}{2}\right)+f\left(a+\frac{h}{2}+h\right)+...+f\left(a+\frac{h}{2}+(n-1)\cdot h\right)\right),$$

а формула трапецій

$$\int_a^b f(x)dx \approx h\cdot\left(\frac{f(a)}{2}+f(a+h)+f(a+2h)+...+f(a+(n-1)\cdot h)+\frac{f(a+n\cdot h)}{2}\right).$$

У цих формулах n – це кількість рівних частин, на які розбивається відрізок [a,b], а h=(b-a)/n. При створенні програми покласти n=20 або більше.

1. $\int_{-1}^{2}\frac{x}{\left(x^2+1\right)^2}dx$;
2. $\int_0^{1/2}4\cos^2 x\,dx$;
3. $\int_0^{\pi/3}\frac{dx}{\cos^2 x}$;
4. $\int_4^9\frac{x+1}{\sqrt{x}}dx$;
5. $\int_{-\pi}^{\pi}\sin 7x\cdot\cos 5x\,dx$;
6. $\int_0^{\pi/2}\frac{\sin 9x}{2+\sin x}dx$;
7. $\int_0^{\pi/2}\sin^6 x\,dx$;
8. $\int_0^{\pi}\frac{x\cdot\sin x}{1+\cos^2 x}dx$;
9. $\int_0^1\frac{\ln(1+x)}{1+x^2}dx$;
10. $\int_1^2\frac{\operatorname{arctgn} x}{x}dx$;
11. $\int_0^{\pi/2}\sqrt{1-\frac{1}{2}\sin^2 x}\,dx$;
12. $\int_0^1 e^{-x^2}dx$;
13. $\int_0^{\pi}\ln(5-4\cos x)dx$;
14. $\int_0^5\left(25-x^2\right)^3dx$;
15. $\int_0^4\sqrt{16-x^2}dx$;
16. $\int_0^{\pi/2}\sin^2 x\cdot\cos^2 x\,dx$;
17. $\int_0^1(1-x)^3x^7dx$;
18. $\int_1^2 x^5\ln^3 x\,dx$.

♖ Додаткові задачі

1. Візьмемо будь-яке тризначне число n. Утворимо з нього 24 числа, виконавши всі перестановки його цифр (включаючи тотожну перестановку). Знайдемо тепер число m, яке рівне середньоарифметичному цих 24 чисел. Вияснити, чи може виконуватися рівність n=m. Якщо так, то знайти всі такі числа n.
2. Знайти чотиризначне число, яке є точним квадратом і у якого дві перші та дві останні цифри однакові.
3. Клієнт винен банку n гривень. Кожний місяць банк збільшує його борг на k% від поточної суми боргу. Через скільки місяців борг клієнта перед банком перевищить m гривень? Написати програму, яка за даними натуральними числами n, k, m виводить відповідь на поставлене питання.
4. Нехай клієнт хоче взяти в банку позику на суму n=1000 гривень під виплачувані проценти k=15% терміном на m=18 місяців. Сума виплачуваних процентів за рік обчислюється за формулою n*k/100=150 гривень. Повна сума виплачуваних процентів за всі роки обчислюється так: s=(n*k/100)*(m/12)=150*1.5=225 гривень. Ця сума одразу забирається із суми позики, і клієнт реально отримує n1=n-s=775 гривень. Виплачувати суму позики клієнт повинен щомісячно рівними частками p=n/m=55.56 гривень.

   Написати програму, яка буде робити запит на введення трьох значень: реальна сума n1, яку хоче отримати клієнт; встановлені банком річні проценти k; термін виплати позики у місяцях m. Програма повинна обчислювати повну суму позики n, щомісячні виплати p, повторювати обчислення стільки разів, скільки хоче користувач.

5. Написати програму, що визначає, якими монетами краще видавати здачу певної суми від 1 до 99 копійок. Використовуються монети вартістю 25, 10, 5, 2, 1 копійок.
6. Набір трьох натуральних чисел x, y, z, для яких виконується рівність $x^2+y^2=z^2$, називають піфагоровими трійками. Знайти всі піфагорові трійки, що не перевищують 500.

Питання для самоконтролю

1) У яких випадках використовують цикл?
2) Які оператори циклу є в мові С?
3) Записати загальний вигляд циклу з параметром. Як він працює?
4) Записати загальний вигляд циклу з передумовою. Як він працює?
5) Записати загальний вигляд циклу з після умовою. Як він працює?
6) Чим цикл ПОКИ відрізняється від ДО?
7) Скільки разів виконається тіло циклу: for i:= 8 to 5 do s :=s+i; ?
8) Яким буде значення параметра **i** після завершення циклу з пункту 7?
9) Скільки разів виконається тіло циклу наступних операторів?
   a) for (i=5; i<=5; i++) s=s+i;
   b) for (i=-5; i<=5; i++) s=s+i;
   c) for (i=5; i<=8; i++) s=s+i;
   d) for (i=8; i<=5; i--) s=s+i;
   e) for (i=5; i<=5; i--) s=s+i;
   f) for (i=-8; i<=5; i--) s=s+i;
10) Чи можна в тілі циклу з параметром змінювати початкове або кінцеве значення параметра?
11) Чи можна в тілі циклу з параметром змінювати значення параметра? Наприклад, for (i=5; i<8; i++) i+=2;
12) Дані якого типу можна використати як лічильник циклу **FOR**?
13) Чи можливі наступні оператори циклу? Відповідь обґрунтуйте. Якщо оператор правильний, то скільки разів виконається тіло циклу?
   a) for (i='a'; i<='z'; i++) s+=i;
   b) for (i=5; i<8; i++) ;
   c) for (i=false; i<=true; i++) s+=i;
   d) for (i=true; i<=false; i--) s+=i;
   e) while (2<5) s=2;
   f) while (2<0) s=2;
   g) while (0.5) s=2;
   h) while (0) s=2;
   i) while (2<5) ;
   j) while s=2;
   k) while ;
   l) while (s=2) 2<0;
   m) do x=2; while (2<5);
   n) do x=2; while (2<0);
   o) do while (2<5);
   p) do while (2<0);
   q) do while (2.5);
   r) do while (0);
   s) do while;
   t) do false while true;

Практичне заняття № 4

Тема: ***Перелікові типи***

Студент повинен знати: прості типи даних, операції над даними цих типів, синтаксис і семантику оператора перемикача.

Теоретичні відомості

Переліковий тип (переліки) визначається як упорядкований набір ідентифікаторів, заданих шляхом їх перерахування. Він створюється користувачем. У мові С переліки – це набори цілочислових констант, кожна з яких має унікальне ім'я. Цей тип оголошується за синтаксисом

enum <тег_переліку> {<список_ідентифікаторів>},

де **enum** – службове слово; <тег_переліку> - ідентифікатор, який служить назвою переліку; <список_ідентифікаторів> - набір імен, які надаються константам переліку. Наприклад, імена тижня можна задати переліком

enum days {Mon, Tue, Wed, Thu, Fri, Sat, Sun};

Кожний елемент переліку є константою типу int, йому присвоюється значення, на одиницю більше, ніж значення попереднього елемента. Стандартно перший елемент переліку отримує значення 0. У наведеному прикладі константа Mon набуває значення 0, Tue – значення 1, і так далі, Sun – значення 6.

Стандартні значення констант можна змінити, присвоївши їм потрібні значення. Наприклад,

enum imena {Petro=2, Ivan, Serhiy=5, Mykola};

Тоді константа Petro матиме значення 2, Ivan – 3, Serhiy – 5, Mykola – 6.

Оскільки константи переліків мають тип int, то вони сумісні з усіма арифметичними типами мови С. Застосування переліків підвищує наочність програми. Можна оголошувати змінні перелікового типу. Значеннями таких змінних є цілі числа, присвоєні їм через іменовані константи. При виведенні значень змінних виводяться цілі числа, а не самі константи. Компілятор не контролює діапазон значень перелікової змінної.

Приклад 1

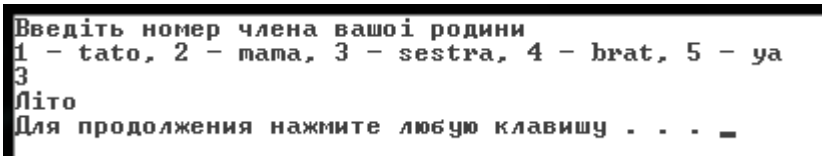
За введеним із клавіатури номером одного з членів вашої родини вивести повідомлення про те, у якій порі року він народився.

Аналіз задачі

Аналіз очевидний, а алгоритм легко зрозуміти з тексту програми.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        enum famili {tato=1, mama, sestra, brat, ya};        // Переліковий тип
        enum famili s;                                       // Оголошення змінної перелікового типу
        int n;                                               // Номер члена родини
        printf ("Введіть номер члена вашої родини\n");
        printf ("1 - tato, 2 - mama, 3 - sestra, 4 - brat, 5 - ya\n");
        scanf("%d", &n);
        switch (n)                                   // За номером визначається значення змінної s
                {       case 1: s=tato; break;
                        case 2: s=mama; break;
                        case 3: s=sestra; break;
```

```
                case 4: s=brat; break;
                case 5: s=ya; break;
                default: {
                                printf ("Невірно введений номер\n");
                                exit(1);
                         }
        }
    switch (s)          // Виведення пори року, у якій народився член родини
        {   case mama:
            case sestra: printf ("Літо\n"); break;
            case tato: printf ("Осінь\n"); break;
            case ya: printf ("Зима\n"); break;
            case brat: printf ("Весна\n");
        }
    system("pause");
    return 0;
}
```



Результат виконання програми.

Задачі

1. Створіть перелікові типи: курс={пн, зх, пд, сх}, наказ={вперед, вправо, назад, вліво}. Змінні К1, К2 мають тип курс, а змінна НК – наказ. Корабель спочатку йшов курсом К1, а потім його курс змінився згідно наказу НК. Визначити К2 – новий курс корабля.
2. Створіть перелікові типи: місяць={січень, лютий, березень, квітень, травень, червень, липень, серпень, вересень, жовтень, листопад, грудень}, деньтижня={понеділок, вівторок, середа, четвер, п’ятниця, субота, неділя}. Оголосити змінні: **d** типу int, значенням якої є число дня місяця, тобто число від 1 до 31; **m** типу місяць; **wd1** і **wd** типу деньтижня. У не високосному році 1січня припадає на день тижня wd1. Визначити:
    1) **wd** – день тижня, який припадає на дату **d**, **m**;
    2) **k** – кількість понеділків у році, які припадають на 13-е число.
3. Гравець кидає два гральні кубики. Грані кубиків містять відповідно 1, 2, 3, 4, 5 і 6 точок. Після того, як кубики зупиняться, обчислюється сума точок на гранях, які видно зверху. Якщо сума, яка випала при першому кидку, виявилася рівною 7 або 11, то виграє гравець. Якщо сума при першому кидку склала 2, 3 або 12, то гравець програє (виграє казино). Якщо сума першого кидка рівна 4, 5, 6, 8, 9 або 10, то ця сума стає «очком» гравця. Щоб виграти, він повинен продовжити кидати кубики до тих пір, поки не набере своє очко ще раз. Гравець програє, якщо при черговому кидку випадає 7. Написати програму, яка моделює гру в кубики.

☺ Індивідуальні завдання

Основний рівень

1. Створіть перелікові типи: сезон={зима, весна, літо, осінь}, місяць={січень, лютий, березень, квітень, травень, червень, липень, серпень, вересень, жовтень, листопад, грудень}. Змінна **m** має тип місяць, а **s** – сезон. Визначити **s** – пору року, на яку припадає місяць **m.**

2. Створіть перелікові типи: країна={Італія, Катар, Китай, Канада, Польща}, континент={Азія, Америка, Європа}. Змінна **s** має тип країна, а **c** – континент. За **s** - назвою країни визначити **c** - назву її континенту.
3. Створіть перелікові типи: країна={Австрія, Болгарія, Греція, Італія, Норвегія, Франція, Німеччина}, столиця={Відень, Софія, Афіни, Рим, Осло, Париж, Бон}. Змінна **st** має тип країна, а **cap** – столиця. За значенням змінної **st** (назва країни) присвоїти змінній **cap** назву столиці цієї країни.
4. Створіть переліковий тип колір = {чорний, сірий, білий} та змінну **c** цього типу. Надрукувати іменовану константу, яка буде присвоєна змінній **c.**
5. Створіть перелікові типи: місто={Кіровоград, Київ, Санкт–Петербург, Москва, Кишинів, Одеса}, країна={Україна, Молдова, Росія}. Змінна **s** має тип місто, а **c** – країна. За **s** - назвою міста визначити **c** - назву країни.
6. Створіть два перелікові типи для списків констант: {ада, бейсік, модула2, лісп, паскаль, пл1, фортран}, {ada, basic, modula2, lisp, pascal, pl1, fortran}. Нехай **P** – змінна першого типу, а **C** – змінна другого типу. За значенням **P** - українською назвою мови програмування присвоїти змінній **C** англійську назву цієї мови.
7. Для цілого числа **k** від 1 до 99 надрукувати фразу ”мені **k** років”, враховуючи при цьому, що при деяких значеннях **k** слово “років” потрібно замінити на слово “роки” або “рік”.
8. Для натурального числа **k** надрукувати фразу “ми знайшли **k** грибів у лісі”, узгодивши закінчення слова “гриб” із числом **k.**
9. Створіть переліковий тип назва={нуль, один, два, три, чотири, п′ять}. Нехай **n** – змінна цього типу, а змінна **d** може приймати значення від 0 до 5. За цифрою **d** присвоїти змінній **n** назву цієї цифри.
10. Створіть переліковий тип день={понеділок, вівторок, середа, четвер, п’ятниця, субота, неділя} і нехай **c** – змінна цього типу. Надрукувати значення змінної **c.**
11. Створіть переліковий тип місяць={січень, лютий, березень, квітень, травень, червень, липень, серпень, вересень, жовтень, листопад, грудень} і нехай **s** – змінна цього типу. За введеним із клавіатури номером місяця **s** вивести на екран повідомлення про те, хто з ваших родичів народився у цьому місяці.
12. Створіть перелікові типи: нота={до, ре, мі, фа, соль, ля, сі}, інтервал={секунда, терція, кварта, квінта, секта, септима}. Змінні **n1, n2** мають тип нота, а **i** – інтервал. Визначити **i** – інтервал, утворений нотами **n1** і **n2** (**n1** ≠ **n2**): секунда – це інтервал із 2-ох сусідніх нот (по колу: наприклад, ре і мі, сі і до), терція – це інтервал через ноту (наприклад, фа і ля, сі і ре).
13. Значеннями змінної **k** є цифри від 1 до 9. Надрукувати значення змінної **k** римськими цифрами.
14. Створіть переліковий тип letter={a, b, c, d} і нехай **x** – змінна цього типу. Ввести значення типу letter (тобто, a, b, c або d) і присвоїти його змінній **x.**
15. Створіть перелікові типи: відмінок={наз, род, дав, знах, оруд, місц}, слово={степ, біль, зошит, стіна}. Змінна **w** має тип слово, а **p** – відмінок. Вивести слово **w** у відмінку **p** в однині (наприклад, якщо **w**=степ і **p**=оруд, то треба надрукувати слово степом).

Підвищений рівень

Створіть перелікові типи:
сезон={зима, весна, літо, осінь};
місяць={січень, лютий, березень, квітень, травень, червень, липень, серпень, вересень, жовтень, листопад, грудень};
день={понеділок, вівторок, середа, четвер, п′ятниця, субота, неділя};

1. Дано дату у вигляді числа і місяця (наприклад, 12 травня), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які настануть після даної дати через **k** днів. Вважати рік не високосним.

2. Дано дату у вигляді числа і місяця (наприклад, 12 травня) і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які були за **k** днів перед даною датою. Вважати рік не високосним.
3. Дано дату у вигляді числа і місяця (наприклад, 12 травня). Відомо, що 1 січня була неділя. Визначити назву дня і сезону, що припав на дану дату. Вважати рік не високосним.
4. Дано дві дати у вигляді числа і місяця (наприклад, 12 травня і 7 серпня). Відомо, що на другу дату припала середа. Визначити назву дня, що припав на першу дату, і назви сезонів, що припали на обидві дати. Вважати рік не високосним.
5. Відомо, що 1 травня була субота. Підрахувати, скільки субот припадає на кожний місяць і вивести цю інформацію на екран. Вважати рік не високосним.
6. Відомо, що 1 травня була неділя. Підрахувати, скільки субот припадає на кожний сезон, і вивести цю інформацію на екран. Вважати рік не високосним.
7. Дано дату у вигляді числа і місяця (наприклад, 12 травня), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які настануть після даної дати через **k** місяців. Вважати рік не високосним.
8. Дано дату у вигляді числа і місяця (наприклад, 12 травня), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які були за **k** місяців перед даною датою. Вважати рік не високосним.
9. Дано дату у вигляді числа і місяця (наприклад, 12 травня), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які настануть після даної дати через **k** сезонів. Вважати рік не високосним.
10. Дано дату у вигляді числа і місяця (наприклад, 12 травня), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які були за **k** сезонів перед даною датою. Вважати рік не високосним.
11. Дано дату у вигляді числа і місяця (наприклад, 12 травня), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які настануть після даної дати через **k** років. Вважати рік не високосним.
12. Дано дату у вигляді числа і місяця (наприклад, 12 травня), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які були за **k** років перед даною датою. Вважати рік не високосним.
13. Дано дату у вигляді числа, місяця і години (наприклад, 12 травня, 9 година), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які настануть після даної дати через **k** годин. Вважати рік не високосним.
14. Дано дату у вигляді числа, місяця і години (наприклад, 12 травня, 10 година), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця і сезону, які були за **k** годин перед даною датою. Вважати рік не високосним.
15. Дано дві дати у вигляді числа, місяця і року (наприклад, 12 травня 2001 року і 7 серпня 2006 року). Вивести назви сезонів, що припадають на дані дати, і підрахувати, через скільки днів після першої дати настане друга дата. Вважати роки не високосними.
16. Дано два сезони і роки (наприклад, літо 2001 року і осінь 2006 року). Підрахувати, через скільки сезонів і років після першого сезону настане другий сезон. Вважати роки не високосними.
17. Дано дату у вигляді числа, місяця і року (наприклад, 12 травня 2000 року), і ця дата припала на день тижня – четвер. Крім того, дано натуральне число **k**. Вивести назву дня, місяця, сезону і рік, які настануть після даної дати через **k** днів. Вважати роки не високосними.

♖ Додаткові задачі

1. У давньояпонському календарі був прийнятий 60-річний цикл, який складався з 12-річних підциклів. Підцикли позначались назвами кольору: зелений, червоний, жовтий, білий і чорний . Усередині кожного підциклу роки носили назви тварин: щура, корови, тигра, зайця, дракона, змії, коня, вівці, мавпи, півня, собаки і свині. (1984 рік – рік зеленого щура – був початком наступного циклу). Написати програму, яка вводить номер деякого року нашої ери і друкує його назву за давньояпонським календарем.
2. Нехай значення функції f(n) дорівнює кількості літер у записі числа **n** українськими словами f(1)=4(один) , f(3)=3 (три), f(42)=8 (сорок два) і т.д. Надрукувати всі натуральні числа **n** менші 100, для яких f(n)=n.

☝ Питання для самоконтролю

1) Як задається переліковий тип?
2) Як оголосити у програмі змінну перелікового типу? Які способи оголошень ви знаєте?
3) Чи можна застосовувати операції введення-виведення до значень перелікового типу?
4) Які операції є допустимими для даних перелікових типів?
5) Створіть свій власний переліковий тип. Яких значень може набувати змінна цього типу?
6) Які з перелікових типів створені правильно:
   a) enum Imena (Іван, Петро, Данило);
   b) enum Imena {Ivan, Petro, Danylo};
   c) enum Imena={Ivan, Petro, Danylo};
   d) enum Імена {Ivan, Petro, Danylo};
   e) enum Imena {“Ivan”, “Petro”, “Danylo”};
   f) enum Imena {Ivan=1, Petro, Danylo};
   g) enum Imena {Ivan=1, Petro, Danylo=1};
   h) enum Imena {Ivan-2, Petro=4, Danylo=6};
   i) enum Dni {Ponedilok..Nedila};
   j) enum Dni {Ponedilok,...,Nedila};
   k) enum Dni {1,2,3,4,5,6,7};
   l) enum Dni {D1,D2,D3,D4,D5,D6,D7};
7) Синтаксис оператора перемикача.
8) Семантика оператора перемикача.

Практичне заняття № 5

Тема: Функції

Студент повинен знати: синтаксичну структуру функцій, синтаксис формальних параметрів, принципи встановлення відповідності між формальними та фактичними параметрами, форми передачі інформації з програми у підпрограму та навпаки, синтаксис виклику підпрограми у програмі, правила локалізації, стандартні функції та їх призначення.

Теоретичні відомості

Підпрограми є основними будівельними блоками, із яких складається програма. У мові С підпрограми реалізуються у вигляді функцій. Їх структура має вигляд

<тип_функції> <ім'я_функції>(<список_формальних_параметрів>)
{
<тіло_функції>
}

і називається означенням фукції. Перший рядок структури функції називається її заголовком. Тіло функції зазвичай складається з оголошень та означень різних об'єктів та операторів. <тип_функції> визначає тип значення, яке повертає функція в точку виклику. <ім'я_функції> - це ідентифікатор. <список_формальних_параметрів> складається із перерахованих через кому формальних параметрів, кожний із яких описується за синтаксисом

<тип_параметра> <ім'я_параметра>

Для функцій важливе значення має тип **void**. Якщо функція має тип **void**, то вона не повертає в точку виклику ніякого значення. У цьому випадку вона аналогом підпрограми, яку прийнято називати процедурою. Якщо список формальних параметрів функції порожній, то його позначають теж через **void.**

Якщо функція повертає в точку виклику якесь значення, то в її тілі повинен зустрічатися хоч один раз оператор **return**, який має синтаксис

return <вираз>

Тип <виразу> повинен співпадати з типом функції. Виконання оператора **return** завершує роботу функції.

Виклик функції має вигляд і є первинним виразом

<ім'я_функції>(<список_фактичних_параметрів>)

Заголовок функції визначає її інтерфейс, тобто містить необхідну інформацію для її активізації (виклику). Важливу роль при цьому відіграють формальні та фактичні параметри. При виклику підпрограми фактичні параметри зіставляються з формальними, і відповідність між ними позиційна. Через параметри інформація передається із програми у підпрограму і навпаки. У мові С реалізований тільки один вид передачі параметрів, який називається передачею за значенням. Через такий вид параметрів здійснюється передача інформації тільки із програми у підпрограму (функцію). Якщо ж потрібно передавати інформацію як із програми у підпрограму, так і навпаки, то формальними параметрами обирають вказівники.

Оголошення функції складається тільки з її заголовку, який називаються прототипом функції.

Приклад 1

Побудувати таблицю значень функції на вказаному інтервалі з даним кроком, створивши функцію користувача.

$$y=\begin{cases} 2\cdot 3^{x} & ,x\le 0;\\ 2\cos x, & 0<x<\pi;\\ -2 & ,x\ge \pi. \end{cases} \qquad \left[-\frac{\pi}{2};\frac{3\pi}{2}\right], \qquad \Delta x=\frac{\pi}{10}.$$

Аналіз задачі

Задача є досить простою. Розглянемо організацію підпрограми – функції. Вона повинна обчислювати **y** за вказаною формулою при відповідних значеннях аргументу **x**. Схематично це можна зобразити на такому малюнку

Вхідними даними для функції є значення **x**, а результатом роботи є значення **y**. Заголовок функції матиме вигляд

double Func(double a).

Функція має один параметр **a** типу double, через який передаватимуться із програми у функцію значення **x**. Вона повертає результат теж double, тобто значення **y**. Її ім’я Func.

Довжина відрізка, на якому потрібно обчислювати значення Func, дорівнює 2π. Він складається із 20 відрізків Δx=π/10. Оскільки також потрібно обчислювати значення **y** в початковій точці, відрізка -π/2, то **y** треба обчислити у 21 точці. Значення x будемо рахувати за формулою -π/2+i*Δx, i=0,1,…,20.

Недоліком такої програми є 21 викликів Func.

```
#include "stdafx.h"
#include "math.h"
#include "windows.h"
#include "iostream"
#define PI 3.1415                       // Число пі
double Func(double a);                  // Прототип функції
int _tmain(int argc, _TCHAR* argv[])
{
        setlocale (LC_ALL,"RUS");
        int i;                          // Параметр циклу
        double x,y;                     // Оголошеня змінних x та y
        // Побудова каркасу таблиці
        printf ("Таблиця значень функції\n");
        for (i=1; i<=25; i++)
                printf ("_");
        printf ("\n");
        printf ("|    X    |    Y    |\n");
        for (i=1; i<=25; i++)
                printf ("_");
        printf ("\n");
        for (i=0; i<=20; i++)
        {
                x=-PI/2+i*PI/10;                // Обчислення x
                y=Func(x);                      // Виклик Func для обчислення y
                printf ("|  %7.4lf  |  %7.4lf  |\n", x, y);         // Виведення x та y
        }
        for (i=1; i<=25; i++)
```

```
            printf ("_");
        printf ("\n");
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------
double Func(double a)                   // Повний опис функції
{
        if (a<=0)
                return 2*exp(a*log(3.));
        else
                if ((a>0)&&(a<PI))
                        return 2*cos(a);
                else
                        return -2;
}
```

```
Таблиця значень функції
┌──────────┬──────────┐
¦    X     ¦    Y     ¦
├──────────┼──────────┤
¦ -1,5708  ¦  0,3561  ¦
¦ -1,2566  ¦  0,5029  ¦
¦ -0,9425  ¦  0,7102  ¦
¦ -0,6283  ¦  1,0029  ¦
¦ -0,3141  ¦  1,4163  ¦
¦  0,0000  ¦  2,0000  ¦
¦  0,3142  ¦  1,9021  ¦
¦  0,6283  ¦  1,6181  ¦
¦  0,9425  ¦  1,1756  ¦
¦  1,2566  ¦  0,6181  ¦
¦  1,5708  ¦  0,0001  ¦
¦  1,8849  ¦ -0,6179  ¦
¦  2,1991  ¦ -1,1755  ¦
¦  2,5132  ¦ -1,6179  ¦
¦  2,8274  ¦ -1,9021  ¦
¦  3,1415  ¦ -2,0000  ¦
¦  3,4557  ¦ -2,0000  ¦
¦  3,7698  ¦ -2,0000  ¦
¦  4,0840  ¦ -2,0000  ¦
¦  4,3981  ¦ -2,0000  ¦
¦  4,7123  ¦ -2,0000  ¦
```

Результат роботи програми.

Приклад 2

Розв’язати біквадратне рівняння, використовуючи підпрограму розв’язання квадратного рівняння.

Аналіз задачі

Біквадратне рівняння має вигляд $ax^4+bx^2+c=0$. Заміною $z=x^2$ воно зводиться до квадратного рівняння $az^2+bz+c=0$. Обчислюємо дискримінант $D=b^2-4ac$.

Якщо $D<0$, то біквадратне рівняння немає дійсних коренів.

Якщо $D=0$, то розв’язком квадратного рівняння є $z=-b/(2a)$. При $z<0$ біквадратне рівняння немає дійсних коренів, при $z=0$ – має один корінь 0, а при $z>0$ – два корені $x_1=\sqrt{z}$, $x_2=-\sqrt{z}$.

Якщо $D>0$, то квадратне рівняння має два корені: $z_1=(-b+\sqrt{D})/(2a)$, $z_2=(-b-\sqrt{D})/(2a)$. Далі потрібно розглянути випадки.

1. Якщо $z_1<0$ і $z_2<0$, то біквадратне рівняння немає дійсних коренів.
2. Якщо $z_1\geq 0$ і $z_2<0$, то біквадратне рівняння має коренем 0 при $z_1=0$ або два корені $x_1=\sqrt{z_1}$, $x_2=-\sqrt{z_1}$ при $z_1>0$.
3. Якщо $z_1<0$ і $z_2\geq 0$, то біквадратне рівняння має коренем 0 при $z_2=0$ або два корені $x_1=\sqrt{z_2}$, $x_2=-\sqrt{z_2}$ при $z_2>0$.

4. Якщо $z_1 \geq 0$ і $z_2 \geq 0$, то біквадратне рівняння має три або чотири корені. При $z_1=0$ або $z_2=0$ – три, а при $z_1>0$ і $z_2>0$ – чотири.

Модель функції для розв'язування квадратного рівняння має вигляд

Вхідними даними для функції є коефіцієнти a, b, c квадратного рівняння. Після роботи функції у програму потрібно повернути корені z_1, z_2 рівняння та їх кількість k. Значення k=0 відповідає випадку, коли квадратне рівняння немає коренів, k=1 – має один корінь, k=2 – має два корені. Величину k повернемо оператором return. Корені z_1, z_2 можна передати у програму через параметри. Відповідні параметри повинні бути вказівниками. Таким чином, заголовок функції матиме вигляд

int KvadRivn(double a, double b, double c, double *z1, double *z2).

```
#include "stdafx.h"
#include "math.h"
#include "windows.h"
#include "iostream"

// Прототип функції розв'язання квадратного рівняння
int KvadRivn(double a, double b, double c, double *z1, double *z2);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int k;                          // Кількість дійсних коренів квадратного рівняння
        double a,b,c,z1,z2;             // a,b,c – коефіцієнти біквадратного рівння, z1,z2 –
                                                корені квадратного рівняння
        printf ("Введіть коефіцієнти a, b, c біквадратного рівняння\n");
        scanf("%lf%lf%lf", &a, &b, &c);
        k=KvadRivn(a, b, c, &z1, &z2);      // Виклик функції розв'язування квадратного
                                                рівняння
        if (k==0)                       // Випадок D<0
        {       printf ("Рівняння немае дійсних коренів\n");
                system("pause");
                return 0;
        }
        if (k==1)                       // Випадок D=0
                if (z1<0)
                {       printf ("Рівняння немае дійсних коренів\n");
                        system("pause");
                        return 0;
                }
                else
                        if (z1==0)
                        {       printf ("Корінь рівняння = %lf\n", z1);
                                system("pause");
                                return 0;
                        }
```

```
                else
                {       printf ("Корені рівняння: x1=%lf    x2=%lf\n", sqrt(z1), -sqrt(z1));
                        system("pause");
                        return 0;
                }
if (k==2)               // Випадок D>0
{       if (z1<0)
                if (z2<0)
                {       printf ("Рівняння немае дійсних коренів\n");
                        system("pause");
                        return 0;
                }
                else
                        if (z2==0)
                                {       printf ("Корінь рівняння = %lf\n", z2);
                                        system("pause");
                                        return 0;
                                }
                        else
                                {       printf ("Корені рівняння: x1=%lf    x2=%lf\n",
                                                                        sqrt(z2), -sqrt(z2));
                                        system("pause");
                                        return 0;
                                }
        if (z1==0)
                if (z2<0)
                {       printf ("Корінь рівняння = %lf\n", z1);
                        system("pause");
                        return 0;
                }
                else
                        if (z2==0)
                                {       printf ("Корінь рівняння = %lf\n", z1);
                                        system("pause");
                                        return 0;
                                }
                        else
                                {       printf ("Корені рівняння: x1=%lf    x2=%lf
                                                        x3=%lf\n", z1, sqrt(z2), -sqrt(z2));
                                        system("pause");
                                        return 0;
                                }
        if (z1>0)
                if (z2<0)
                {       printf ("Корені рівняння: x1=%lf    x2=%lf\n", sqrt(z1), -sqrt(z1));
                        system("pause");
                        return 0;
                }
                else
                        if (z2==0)
                                {       printf ("Корені рівняння: x1=%lf    x2=%lf
                                                x3=%lf\n", z2, sqrt(z1), -sqrt(z1));
```

```
                                        system("pause");
                                        return 0;
                                }
                        else
                        {       printf ("Корені рівняння: x1=%lf    x2=%lf
                                x3=%lf   x4=%lf\n", sqrt(z1), -sqrt(z1), sqrt(z2), -sqrt(z2));
                                system("pause");
                                return 0;
                        }
        }
}
//-------------------------------------------------------------------------------------------------------------------
int KvadRivn(double a, double b, double c, double *z1, double *z2)
{       double D;
        D=b*b-4*a*c;                            // Обчислення дискримінанта
        if (D<0)                                // Дійсних коренів немає
                return 0;                       // Повертаємо кількість коренів
        else
                if (D==0)                       // Рівняння має один корінь
                {       *z1=-b/(2*a);           // Обчислюємо корінь
                        return 1;               // Повертаємо кількість корінь
                }
                else                            // Рівняння має два корені
                {       *z1=(-b+sqrt(D))/(2*a); // Обчислюємо корені
                        *z2=(-b-sqrt(D))/(2*a);
                        return 2;               // Повертаємо кількість коренів
                }
}
```

```
Введіть коефіцієнти a, b, c біквадратного рівняння
1 -13 36
Корені рівняння: x1=3,000000    x2=-3,000000    x3=2,000000    x4=-2,000000
Для продолжения нажмите любую клавишу . . .
```

```
Введіть коефіцієнти a, b, c біквадратного рівняння
1 3 2
Рівняння немае дійсних коренів
Для продолжения нажмите любую клавишу . . .
```

```
Введіть коефіцієнти a, b, c біквадратного рівняння
1 0 -16
Корені рівняння: x1=2,000000    x2=-2,000000
Для продолжения нажмите любую клавишу . . .
```

```
Введіть коефіцієнти a, b, c біквадратного рівняння
1 2 0
Корінь рівняння = 0,000000
Для продолжения нажмите любую клавишу . . .
```

Приклади розв’язання задачі.

Задачі

1. Дано довжини сторін п’ятикутника і дві його діагоналі, що виходять з одної вершини. Обчислити площу п’ятикутника.
2. Дані дійсні числа a, b, c. Обчислити

$$\frac{\max(a, a+b)+\max(a, b+c)}{1+\max(a+bc, 115)}$$

3. З клавіатури вводяться n довільних символів. Кожний символ має свій двійковий код. Обчислити скільки нулів та одиниць використано для кодування цих n символів. Вивести на екран кожний символ, його двійковий код, кількість нулів в коді, кількість одиниць в коді. Виведення оформити у вигляді таблиці.
4. Дана послідовність n натуральних чисел. Вивести у вигляді таблиці число, кількість в його записі цифр: 0, 1,…, 9.

☺ Індивідуальні завдання

Основний рівень

А

Побудувати таблицю значень функції на вказаному інтервалі з даним кроком, створивши функцію користувача.

1. $y=\begin{cases} -x, x\le 0; \\ x^2, 0<x<1; \\ x^3, x\ge 1. \end{cases}$ $[-1;2]$, $\Delta x=0.1$;
2. $y=\begin{cases} 1, x\le 0; \\ \cos x, 0<x<\pi; \\ -1, x\ge \pi. \end{cases}$ $\left[-\frac{\pi}{2};\frac{3\pi}{2}\right]$, $\Delta x=\frac{\pi}{10}$;
3. $y=\begin{cases} -1, x\le \frac{\pi}{2}; \\ \sin x, -\frac{\pi}{2}<x<\frac{\pi}{2}; \\ 1, x\ge \frac{\pi}{2}. \end{cases}$ $[-\pi;\pi]$, $\Delta x=\frac{\pi}{10}$;
4. $y=\begin{cases} \operatorname{tg} x, -\frac{\pi}{4}<x<\frac{\pi}{4}; \\ \frac{4x}{\pi}, \text{при інших } x. \end{cases}$ $[-\pi;\pi]$, $\Delta x=\frac{\pi}{10}$;
5. $y=\begin{cases} x, x\le 1; \\ 1, 1<x<3; \\ -x+4, x\ge 3. \end{cases}$ $[0;4]$, $\Delta x=0.25$;
6. $y=\begin{cases} \operatorname{ctg} x, \frac{\pi}{4}<x<\frac{3\pi}{4}; \\ -\frac{4x}{\pi}+2, \text{при інших } x. \end{cases}$ $[0;\pi]$, $\Delta x=\frac{\pi}{20}$;
7. $y=\begin{cases} x^2, x\le 0; \\ 0, 0<x<1; \\ \ln x, x\ge 1. \end{cases}$ $[-2;8]$, $\Delta x=0.25$;

8. $y=\begin{cases} -\dfrac{2x}{\pi}-1, x\le -\dfrac{\pi}{2}; \\ \cos x, -\dfrac{\pi}{2}<x<0; \\ 5^x, x\ge 0. \end{cases}$ $[-5;5]$, $\Delta x=0.25$;

9. $y=\begin{cases} |x|, x\le 0; \\ \sqrt{x}, 0<x<4; \\ 2, x\ge 4. \end{cases}$ $[-2;8]$, $\Delta x=0.25$;

10. $y=\begin{cases} |x|, x\le 0; \\ 4\sin 2x, 0<x<\pi; \\ 0, x\ge \pi. \end{cases}$ $\left[-\dfrac{\pi}{2};\dfrac{3\pi}{2}\right]$, $\Delta x=\dfrac{\pi}{10}$;

11. $y=\begin{cases} 2^{-x}, x\le 0; \\ \cos x, 0<x<\dfrac{\pi}{2}; \\ \dfrac{2x}{\pi}-1, x\ge \dfrac{\pi}{2}. \end{cases}$ $\left[-\dfrac{\pi}{2};\dfrac{3\pi}{2}\right]$, $\Delta x=\dfrac{\pi}{10}$;

12. $y=\begin{cases} x-1, x\le 1; \\ \log_2 x, 1<x<4; \\ 2, x\ge 4. \end{cases}$ $[0;5]$, $\Delta x=0.2$;

13. $y=\begin{cases} x-1, x\le 1; \\ \log_{1/2} x, 1<x<4; \\ x-6, x\ge 4. \end{cases}$ $[0;5]$, $\Delta x=0.2$;

14. $y=\begin{cases} x+2, x\le 0; \\ 2\cos 2x, 0<x<\pi; \\ 2, x\ge \pi. \end{cases}$ $\left[-\dfrac{\pi}{2};\dfrac{3\pi}{2}\right]$, $\Delta x=\dfrac{\pi}{10}$;

15. $y=\begin{cases} |x|, x\le 0; \\ 2\sin\dfrac{1}{2}x, 0<x<\pi; \\ 2, x\ge \pi. \end{cases}$ $\left[-\dfrac{\pi}{2};\dfrac{3\pi}{2}\right]$, $\Delta x=\dfrac{\pi}{10}$.

Б

1. Визначити, чи є серед цифр даного тризначного числа однакові. Створити програму, у якій використовується функція для перевірки цифр числа. Функція повинна повертати значення 1 (true) або 0 (false).
2. Визначити, чи рівна сума перших двох цифр даного чотиризначного числа сумі двох його останніх цифр. Створити програму, у якій використовується функція для перевірки рівності сум. Функція повинна повертати значення 1 (true) або 0 (false).
3. Перевірити, чи дорівнює квадрат даного тризначного числа кубу суми цифр цього числа. Створити програму, у якій використовується функція для виконання перевірки. Функція повинна повертати значення 1 (true) або 0 (false).
4. Дано три довільних числа. Перевірити, чи можна побудувати трикутник із такими довжинами сторін. Створити програму, у якій використовується функція для виконання перевірки. Функція повинна повертати значення 1 (true) або 0 (false).

5. Дано координати (як цілі від 1 до 8) двох полів шахової дошки. Перевірити, чи може кінь за один хід перейти з одного поля на друге. Створити програму, у якій використовується функція для виконання перевірки. Функція повинна повертати значення 1 (true) або 0 (false).
6. Дано координати (як цілі від 1 до 8) двох полів шахової дошки. Перевірити, чи може тура за один хід перейти з одного поля на друге. Створити програму, у якій використовується функція для виконання перевірки. Функція повинна повертати значення 1 (true) або 0 (false).
7. Дано координати (як цілі від 1 до 8) двох полів шахової дошки. Перевірити, чи може ферзь за один хід перейти з одного поля на друге. Створити програму, у якій використовується функція для виконання перевірки. Функція повинна повертати значення 1 (true) або 0 (false).
8. Знайти добуток цифр даного чотиризначного числа. Створити програму із використанням функції, яка обчислює добутку цифр і повертає його.
9. Визначити номер чверті координатної площини, у якій знаходиться точка з координатами (x,y), x*y ≠ 0. Створити програму із використанням функції, яка визначає номер чверті і повертає його.
10. За номером y (y>0) деякого року нашої ери визначити номер століття (врахувати, що, наприклад, початком XX століття був 1901 рік, а не 1900). Створити програму із використанням функції, яка визначає номер століття і повертає його.
11. Визначити, скільки разів зустрічається цифра 0 у записі даного чотиризначного натурального числа. Створити програму із використанням функції, яка обчислює і повертає кількість нулів.
12. Сторони квадрата рівні **a** і паралельні осям координат, а його центр міститься в початку координат. Перевірити, чи міститься точка (x,y) всередині квадрата. Створити програму із використанням функції, яка виконує перевірку і повертає значення 1 (true) або 0 (false).
13. Дано дві трійки чисел a_1, b_1, c_1 та a_2, b_2, c_2. Обчислити вираз.
$$s = \frac{\max(a_1, b_1, c_1) + \max(a_2, b_2, c_2)}{2}.$$
14. Дано дві трійки чисел a_1, b_1, c_1 та a_2, b_2, c_2. Обчислити вираз.
$$d = \sqrt{\min(a_1, b_1, c_1) \cdot \min(a_2, b_2, c_2)}.$$
15. Дано довжини сторін a, b, c, d чотирикутника і його діагональ f. Обчислити площу чотирикутника. Створити програму, у якій використовується функція для обчислення площі трикутника за трьома сторонами.

В

1. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – довжини відрізків. Вияснити, які з трійок утворюють трикутники. Підрахувати кількість існуючих трикутників, вивести їх порядкові номери та площі. Використати функцію для перевірки існування трикутника. Якщо трикутник існує, то функція додатково обчислює його площу.
2. Дана послідовність **n** натуральних чисел. Для кожного числа обчислити кількість його цифр та суму цих цифр. Вивести на екран кожне число, кількість його цифр ти їх суму. Використати функцію для обчислення кількості цифр числа та їх суми.
3. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – дійсні числа. У кожній трійці знайти найбільше число і найменше, створивши для цього функцію. Обчислити суму найбільших чисел та суму найменших чисел трійок.
4. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – цілі числа. Підрахувати кількість трійок, у яких хоча б одне число дорівнює нулю, та обчислити суму всіх чисел у таких трійках. Створити функцію, яка перевіряє, чи містить трійка нуль, і якщо містить, то функція обчислює суму чисел трійки.

5. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – цілі числа. Підрахувати кількість трійок, у яких числа a_i, b_i, c_i одного знаку, та обчислити суму всіх чисел у всіх таких трійках. Створити функцію, яка перевіряє, чи числа a_i, b_i, c_i одного знаку, і якщо так, то функція обчислює суму чисел трійки.
6. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – цілі числа. Підрахувати кількість трійок, у яких хоча б одна пара чисел різних знаків, та обчислити суму всіх чисел у таких трійках. Створити функцію, яка виконує перевірку трійок, і якщо трійка має потрібну властивість, то функція обчислює суму її чисел.
7. Дана послідовність **n** натуральних чисел. Вивести на екран всі числа, цифри яких розташовані за зростанням (наприклад, 126, 4679), та обчислити в таких числах різницю між найбільшою і найменшою цифрами. Використати функцію, яка знаходить потрібні числа та обчислює вказану різницю.
8. Дана послідовність **n** натуральних чисел. Вивести на екран всі числа, цифри яких розташовані за спаданням (наприклад, 621, 9743), та обчислити в таких числах різницю між найбільшою і найменшою цифрами. Використати функцію, яка знаходить потрібні числа та обчислює вказану різницю.
9. Дана послідовність **n** натуральних чисел. Вивести на екран всі числа, сума цифр яких дорівнює кількості цифр. Використати функцію, яка знаходить потрібні числа.
10. Дано **n** відрізків, кожний із яких заданий координатами точок (a_i, b_i), (c_i,d_i). Для кожного відрізка визначити, які з осей координат він перетинає. Створити для перевірки відповідну функцію. Відрізок може не перетинати осей координат, або перетинати тільки вісь ОХ чи тільки вісь ОУ, або перетинати обидві вісі. Обчислити скільки разів відрізки перетнуть вісь ОХ та вісь ОУ.
11. Дана послідовність **n** натуральних чисел. Для кожного числа визначити, скільки разів у його записі зустрічається цифра 0 і скільки разів цифра 1. Вивести на екран кожне число, кількість у ньому цифр 0 та цифр 1. Підрахувати загальну кількість окремо нулів та окремо одиниць, що зустрічаються в записах даних **n** чисел. Використати функцію, яка підраховує нулі та одиниці в записі числа.
12. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – цілі числа. Ці трійки поділимо на три види: 1) число b_i більше за своїх сусідів; 2) число b_i менше за своїх сусідів; 3) інші трійки. Підрахувати кількість трійок кожного виду. Створити функцію, яка визначає вид трійки.
13. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – цілі числа. Ці трійки поділимо на три види: 1) зростаючі, тобто $a_i<b_i<c_i$; 2) спадні, тобто $a_i>b_i>c_i$; 3) інші трійки. Підрахувати кількість трійок кожного виду. Створити функцію, яка визначає вид трійки.
14. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – цілі числа. Ці трійки поділимо на три види: 1) рівносторонні, тобто $a_i=b_i=c_i$; 2) рівнобедрені, тобто в трійці є тільки два рівних числа; 3) інші трійки. Підрахувати кількість трійок кожного виду. Створити функцію, яка визначає вид трійки.
15. Дана послідовність **n** трійок (a_i, b_i, c_i), де a_i, b_i, c_i – довжини сторін трикутника. Підрахувати, скільки серед **n** трикутників є рівносторонніх, рівнобедрених і різносторонніх. Створити функцію, яка визначає вид трикутника.

♖ Додаткові задачі

1. Знайти суму кубів всіх коренів рівняння $ex^3 - \pi x^2 - (2e+1)x + 2\pi = 0$, використовуючи підпрограму розв'язання рівняння.
2. Використовуючи процедуру для обчислення степеня числа, знайти значення виразу $y=a_4 \cdot x^4+a_3 \cdot x^3+a_2 \cdot x^2+a_1 x+a_0$. Коефіцієнти a_4, a_3, a_2, a_1, a_0 та x вводяться з клавіатури.

3. Знайти найменше спільне кратне (НСК) двох натуральних чисел, використовуючи обчислення найбільшого спільного дільника (НСД). Вказівка: для будь-яких натуральних чисел a,b справедлива тотожність a×b=НСД(a,b)×НСК(a,b).
4. Знайти найбільший спільний дільник (НСД) трьох натуральних чисел, використовуючи обчислення найбільшого спільного дільника (НСД) двох чисел.
5. Вивести на екран у порядку зростання перші 1000 чисел, які не мають простих дільників, крім 2, 3 і 5. (Початок списку: 2, 3, 4, 5, 6, 8, 9, 10, 12, 15,… .)
6. Вивести на екран усі пари «дружніх» чисел, що не перевищують заданого натурального числа. (Два натуральних числа називаються «дружніми», якщо кожне з них дорівнює сумі всіх дільників іншого за винятком його самого. Наприклад, числа 220 та 284).
7. Використовуючи функцію, знайти всі автоморфні числа з проміжку від A до B. Автоморфним називається число, квадрат якого закінчується ним самим. Наприклад, автоморфним є число 6, тому що його квадрат 36 закінчується на 6, або число 25 – його квадрат 625.
8. Написати програму підрахунку кількості “щасливих” автобусних квитків.
9. Серед чисел з проміжку від A до B знайти всі прості.
10. Дано натуральне число **n**. З’ясувати, чи є серед чисел **n, n+1,…,2n** близнюки, тобто прості числа, різниця між якими дорівнює 2.

Питання для самоконтролю

1. Які види підпрограм можуть бути в мовах програмування?
2. Яка структура функції?
3. Якою повинна бути функція, щоб вона була аналогом процедури?
4. Які види параметрів можуть бути у процедури чи функції?
5. Які параметри називаються формальними, а які – фактичними?
6. У чому відмінність між локальними та глобальними змінними?
7. Якщо функція є аналогом процедури, то як вона викликається?
8. У яких елементах програми можуть зустрічатися виклики функції?
9. Яким чином функція повертає результат своєї роботи?

Практичне заняття № 6

Тема: ***Одновимірні масиви***

Студент повинен знати: означення масиву, синтаксис оголошення масиву та його ініціалізацію, індексацію елементів масиву та виконання дій над ними.

Теоретичні відомості

Одновимірний масив – скінченна послідовність однотипних елементів. З точки зору математики, такий масив є вектором. Масив оголошується за синтаксисом:

<тип_елементів> <ім'я_масиву>[<кількість_елементів>];

У цьому оголошенні <тип_елементів> - довільний допустимий тип мови С простий чи складений, <ім'я_масиву> - ідентифікатор, квадратні дужки [ ] – обов'язкова ознака масиву, <кількість_елементів> - константний вираз, значенням якого є натуральне число.

Кожний елемент масиву має індекс (номер). Індексація починається з нуля, тобто перший елемент має індекс 0. Для масиву компілятор виділяє неперервну ділянку пам'яті розміру **кількість_елементів** x **sizeof(тип_елементів).**

При означенні масиву відбувається його ініціалізація. Застосовують дві форми ініціалізації:

- із зазначенням кількості елементів;
- без зазначення кількості елементів.

Наприклад, у наступних перших двох ініціалізаціях масиву вказана кількість елементів, а в останній – ні.

```
int vector[5] = {-4, 2, 6, 0, 8};
int vector[5] = {-4, 2, 6};
int vector[] = {-4, 2, 6, 0, 8};
```

Елемент масиву є змінною, яка має складне ім'я.

Для доступу до елементів масиву використовують дві форми звертання:

- через індекси;
- через вказівники (адреси).

Звертання до елемента масиву через індекс має синтаксис

<ім'я_масиву>[<індекс_елемента>]

<індекс_елемента> є цілочисловим виразом, який визначає індекс (порядковий номер) елемента. Наприклад, масив int vector[5] складається з елементів vector[0], vector[1], vector[2], vector[3], vector[4]. Усі ці елементи є змінними.

Часто значення елементів масиву вводять з клавіатури. Наприклад, для масиву vector це можна виконати в циклі

```
for (i=0; i<5; i++)   scanf("%d", &vector[i]);
```

Ім'я масиву є константним вказівником на початок масиву в пам'яті, тобто ім'я масиву адресує перший елемент масиву. У наведеній таблиці вказані два рівносильних звертання до елементів масиву vector.

Звертання через вказівники	Звертання через індекси
vector	&vector[0]
*vector	vector[0]
*vector = 4;	vector[0] = 4;
vector+2	&vector[2]
*(vector+2)	vector[2]
*(vector+2) = 5;	vector[2] = 5;
*(vector+i)	vector[i]

Через вказівники можна ввести елемент масиву з клавіатури. Наприклад,

```
for (i=0; i<5; i++)   scanf("%d", vector+i);
```

Аналогічно можна організувати виведення елементів масиву

```
for (i=0; i<5; i++)   printf("%d     ", vector[i]);
for (i=0; i<5; i++)   printf("%d     ", *(vector+i) );
```

Приклад 1

Підрахувати кількість додатних елементів лінійного масиву a[10], який складається з цілих чисел.

Аналіз задачі

Для підрахунку додатних елементів введемо лічильник **k** і початкове значення лічильника покладемо рівним 0. Далі послідовно переглядатимемо елементи масиву **a** і порівнюватимемо їх із нулем. Якщо зустрівся додатній елемент, то лічильник збільшуємо на 1. Після перегляду всього масиву значення лічильника буде рівним шуканому числу додатних елементів.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 10                                    // N – кількість елементів масиву
int _tmain(int argc, _TCHAR* argv[])
{
        setlocale (LC_ALL,"RUS");
        int i;
        int a[N];                               // Оголошення масиву
        printf ("Введіть %d цілих чисел\n", N);
        for (i=0; i<N; i++)                     // Ініціалізація масиву з клавіатури
                scanf("%d", &a[i]);
        int k=0;                                // Початкове значення лічильника
        for (i=0; i<N; i++)                     // Пошук додатних елементів
                if (a[i]>0) k++;
        printf ("Кількість додатних елементів масиву = %d\n", k);
        system("pause");
        return 0;
}
```

```
Введіть 10 цілих чисел
2 -1 0 -2 4 15 7 -8 2 4
Кількість додатних елементів масиву = 6
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 2

Написати програму множення двох многочленів

$$p = p_0 x^n + p_1 x^{n-1} + ... + p_{n-1} x + p_n, \qquad q = q_0 x^m + q_1 x^{m-1} + ... + q_{m-1} x + q_m.$$

Аналіз задачі

Добутком многочленів p і q є многочлен f степеня n+m, який рівний

$$f = p \cdot q = c_0 x^{n+m} + c_1 x^{n+m-1} + ... + c_{n+m-1} x + c_{n+m}.$$

Будемо вважати, що перший многочлен має вищий степінь ніж другий, точніше n≥m. Тоді коефіцієнти c_k многочлена f при k = 0, 1, …, m обчислюються за формулою

$$c_k = p_0 q_k + p_1 q_{k-1} + p_2 q_{k-2} + ... + p_k q_0 = \sum_{i=0}^{k} p_i q_{k-i}, \qquad (1)$$

а при k = 1+m, 2+m, ..., n+m – за формулою

$$c_k = p_s q_m + p_{s+1} q_{m-1} + p_{s+2} q_{m-2} + ... + p_u q_{k-u}. \qquad (2)$$

Для формули (2) k = s+m. У цій формулі коефіцієнти многочлена p змінюються від p_s до p_u, де u – це найбільше натуральне число, яке задовольняє одночасно умови $u \le k$, $u \le n$. За формулою (1) ми обчислимо m+1 коефіцієнтів многочлена f, а за формулою (2) – n коефіцієнтів.

У програмі використаємо три масиви: масив із n+1 елементів для коефіцієнтів многочлена p, масив із m+1 елементів для коефіцієнтів многочлена q, , масив із n+m+1 елементів для коефіцієнтів многочлена f.

Для обчислення добутку двох многочленів створимо функцію, модель якої має вигляд

У функцію потрібно передати інформацію про два многочлени **a** і **b**, а із функції потрібно передати у програму інформацію про многочлен **c** = **a**·**b**. Прототип такої функції матиме вигляд

void DobutokMn(double a[], int n, double b[], int m, double c[], int h);

Ім'я функції DobutokMn, її тип void, тобто вона не повертає ніякого значення в точку виклику. Многочлени **a**, **b**, **c** задаються відповідними масивами коефіцієнтів. Значить у функцію потрібно передати масиви a[n+1], b[m+1], а отримати із функції масив c[n+m+1]. У мові С передача масивів як параметрів функції здійснюється за адресою масиву (через вказівники). Інформація про многочлен **a** передається через два формальні параметри

double a[], int n.

Параметр double a[] є вказівником на масив типу double і через нього передається у функцію масив коефіцієнтів многочлена **a** (ми вважаємо коефіцієнти дійсними числами), а через параметр int n ми передаємо степінь цього многочлена. Аналогічно, параметри double b[], int m передають інформацію про многочлен **b**, а параметри double c[], int h – про **c**.

Виклик функції матиме вигляд

DobutokMn(p, N, q, M, f, N+M).

Тут формальні параметри замінені фактичними. p – ім'я масиву коефіцієнтів многочлена p, N – степінь цього многочлена. У мові С ім'я масиву є константним вказівником на початок цього масиву. Тому значенням p є адреса початку масиву. Аналогічно, q – ім'я масиву коефіцієнтів многочлена q, M – його степінь, f – ім'я масиву коефіцієнтів многочлена f, N+M – його степінь.

Виникає питання, як отримати у програмі масив f (коефіцієнти многочлена f), якщо він обчислюються у функції. Цьому масиву пам'ять виділяється у програмі. Передаючи у функцію адресу масиву і виконуючи у функції обчислення над цим масивом, ми насправді заносимо дані у пам'ять масиву у програмі. Тому після завершення роботи функції у програмі у нас буде вся інформація про масив f. Таким чином, використання адреси в якості фактичного параметра дозволяє передавати інформацію із функції у програму.

Записуючи у прототипі функції DobutokMn формальний параметр у вигляді double a[], ми підкреслюємо, що a є вказівником на тип double, який адресуватиме масив. Це дає можливість звертатися до елементів масиву через індекси. Однак, прототип функції можна було б записати у вигляді

void DobutokMn(double *a, int n, double *b, int m, double *c, int h);

У цьому випадку до елементів масиву потрібно звертатися через вказівники. Код функції DobutokMn для першого варіанту прототипу наведений нижче у програмі. Перепишемо цю функцію для другого варіанту прототипу.

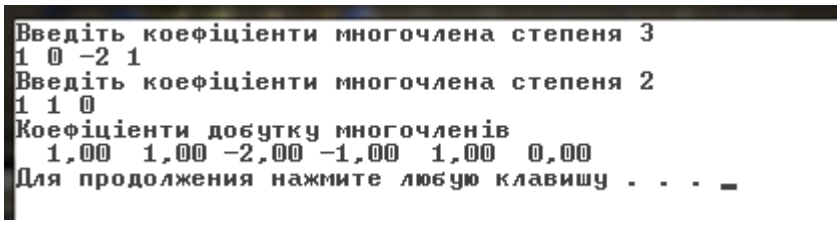
```
void DobutokMn(double *a, int n, double *b, int m, double *c, int h)
```

```
{       int i, k, s;
        for (k=0; k<=m; k++)                    // Обчислення коефіцієнтів за формулою (1)
        {       *(c+k)=0;                       // k-й елемент масиву c
                for (i=0; i<=k; i++)
                *(c+k)+=(*(a+i))*(*(b+k-i));
        }
        for (k=1+m; k<=n+m; k++)                // Обчислення коефіцієнтів за формулою (2)
        {       *(c+k)=0;
                s=k-m;
                while ((s<=k) &&(s<=n))
                {       *(c+k)+=(*(a+s))*(*(b+k-s));
                        s++;
                }
        }
}
```

Далі наведений код програми

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 3                             // Степінь многочлена p
#define M 2                             // Степінь многочлена q
// Прототип функції для обчислення добутку многочленів a і b
void DobutokMn(double a[], int n, double b[], int m, double c[], int h);
int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i;
        double p[N+1];                  // Масив коефіцієнтів многочлена p
        double q[M+1];                  // Масив коефіцієнтів многочлена q
        double f[N+M+1];                // Масив коефіцієнтів многочлена f
        printf ("Введіть коефіцієнти многочлена степеня %d\n", N);
        for (i=0; i<=N; i++)            // Введення коефіцієнтів многочлена p
                scanf("%lf", &p[i]);
        printf ("Введіть коефіцієнти многочлена степеня %d\n", M);
        for (i=0; i<=M; i++)            // Введення коефіцієнтів многочлена q
                scanf("%lf", &q[i]);
        if (N>M)
                DobutokMn(p, N, q, M, f, N+M);          // Виклик функції
        else
                DobutokMn(q, M, p, N, f, N+M);          // Виклик функції
// Виведення результату - коефіцієнти многочлена f
        printf ("Коефіцієнти добутку многочленів\n");
        for (i=0; i<=N+M; i++)
                printf("%6.2lf"  , f[i]);
        printf("\n");
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------
void DobutokMn(double a[], int n, double b[], int m, double c[], int h)
{       int i, k, s;
        for (k=0; k<=m; k++)                    // Обчислення коефіцієнтів за формулою (1)
```

```
        {       c[k]=0;
                for (i=0; i<=k; i++)
                c[k]+=a[i]*b[k-i];
        }
        for (k=1+m; k<=n+m; k++)          // Обчислення коефіцієнтів за формулою (2)
        {       c[k]=0;
                s=k-m;
                while ((s<=k) &&(s<=n))
                {       c[k]+=a[s]*b[k-s];
                        s++;
                }
        }
}
```



Результат роботи програми.

Задачі

1. Обчислити значення многочлена $a_0x^n+a_1x^{n-1}+...+a_{n-1}x+a_n$ з дійсними коефіцієнтами та його похідної в точці x_0. Скористатися схемою Горнера.
2. Дана послідовність елементів із n різних дійсних чисел. Знайти мінімальний та максимальний елементи послідовності та поміняти їх місцями.
3. При киданні грального кубика може випасти від 1 до 6 очок. Нехай зроблено n кидків. Написати програму, яка підраховує скільки разів випало кожне очко.
4. Надрукувати всі цифри десяткового запису чисел 2^{100} і 1!+2!+…+100!.

☺ Індивідуальні завдання

Основний рівень

1. Знайдіть кількість ненульових елементів масиву цілих чисел.
2. Напишіть програму, яка дає відповідь “так” чи “ні” залежно від того, чи зустрічається число 7 у масиві цілих чисел.
3. Дано масив цілих чисел. Знайдіть різницю найбільшого та найменшого чисел.
4. Вхідними даними є цілочислова таблиця “температура”[1..31], у якій записана температура за кожний день січня, і величина s, яка рівна середній температурі в січні за останнє століття. Підрахувати, скільки в січні було днів з температурою більшою, меншою та рівною середній.
5. Дана цілочислова таблиця. Знайти кількість елементів цієї таблиці більших за середнє арифметичне всіх її елементів.
6. Написати програму циклічної перестановки елементів таблиці, при якій A[i] переміщується в A[i+1] для i = 1, 2,…, n, а A[n] переміщується в A[1].
7. Дані дві цілочислові таблиці A і B. Підрахуйте кількість тих i, для яких
   а)A[i]<B[i]; б)A[i]>B[i]; в)A[i]=B[i].
8. Дано масив із 365 елементів, що містить середньодобові температури кожного дня не високосного року. Визначити місяць із найбільшою середньомісячною температурою.
9. N точок задано своїми координатами на площині: масив X[i] містить абсциси точок, масив Y[i] - ординати. Знайти дві точки, відстань між якими найбільша.

10. Дано масив цілих чисел. Визначити, чи є у масиві два нульові елементи, які розташовані підряд.
11. Дано два масиви, що складаються з однакової кількості цілих чисел. Отримати третій масив тієї ж розмірності, кожний елемент якого дорівнює більшому з відповідних елементів даних масивів.
12. Дано одновимірний масив. Переставити у зворотному порядку елементи масиву, які розташовані між мінімальним та максимальним елементами.
13. Дано масив цілих чисел. Знайдіть кількість найбільших елементів масиву та їх індекси.
14. Дано масив цілих чисел. Знайдіть кількість найменших елементів масиву та їх індекси.
15. Дано два вектори із n цілих чисел. Обчислити суму, різницю і скалярний добуток векторів та вивести результати на екран.

Підвищений рівень

1. Дано одновимірний масив B, який складається з 2n елементів. Розташувати елементи масиву в такому порядку:
   B[n+1], B[n+2],…, B[2n], B[1], B[2], …, B[n].
2. B[n+1], B[n+2],…, B[2n], B[n], B[n-1], …, B[1].
3. B[1], B[n+1], B[2], B[n+2],…, B[n], B[2n].
4. B[2n], B[2n-1],…, B[n+1], B[1], B[2],…, B[n].
5. Дано масив із n дійсних чисел. Перетворити цей масив так: усі від'ємні числа перенести в початок масиву, а всі інші – у кінець, зберігаючи початкове взаємне розташування як серед від'ємних, так і серед інших чисел.
6. Дано масив із n дійсних чисел. Упорядкувати його за зростанням.
7. Дано масив із n дійсних чисел. Упорядкувати його за спаданням.
8. Дано масив A з n натуральних чисел. Утворити новий масив B, що містить лише різні елементи масиву A. Упорядкувати масив B за зростанням.
9. Дано дійсні числа a_1,…, a_n. Переставити члени цієї послідовності так, щоб спочатку були розташовані всі її невід'ємні члени, а потім усі від'ємні. Порядок як серед невід'ємних членів, так і серед від'ємних повинен бути збережений.
10. Створити переліковий тип ім'я={Валентина, Євген, Геннадій, Марія, Ніна, Олександр, Тетяна, Федір, Ольга}. За цим типом створити масив Стать, елементи якого приймають значення із множини {чол., жін.}, та масив Зріст, елементи якого приймають значення від 150 до 200. За масивами Стать і Зріст визначити імена найвищих чоловіка і жінки та середній зріст чоловіків і жінок.
11. Створіть перелікові типи: країна={Україна, Росія, Білорусія}, місто={Київ, Москва, Мінськ, Одеса, Миколаїв, Томськ}. Створити масив X із 20 елементів, значеннями яких є константи типу місто. Надрукувати назву країни, міста якої найчастіше зустрічаються в масиві X.
12. Дано два лінійні масиви символів. Знайти найменший серед тих символів першого масиву, який не входять у другий масив.
13. Дано цілі числа x_1, y_1,…, x_n, y_n. З'ясувати, чи є серед точок (x_1, y_1),…, (x_n, y_n) чотири вершини квадрата. Якщо є, то вивести ці вершини.
14. Дано цілочисловий масив. Знайти найменше число елементів, які потрібно відкинути, щоб у масиві залишилася зростаюча послідовність.
15. Дано два впорядковані за зростанням масиви: масив X із n дійсних чисел і масив Y із m дійсних чисел. Об'єднати масиви X і Y в один упорядкований за зростанням масив.
16. Напишіть програму підрахунку перших 30 чисел Фібоначчі (ці числа визначаються так: a[1]=1, a[2]=1, а кожне наступне число обчислюється за формулою: a[n+2]=a[n+1]+a[n]).

♖ Додаткові задачі

1. Дана послідовність a_1, a_2,…,a_n дійсних чисел. Знайти b - середнє арифметичне чисел послідовності і найбільше відхилення від середнього, тобто max(|a_1-b|, |a_2-b|,…,|a_n-b|).
2. Двійковим кодом довжини n називається впорядкована послідовність $a_1a_2…a_n$, де кожне a_i рівне 0 або 1. Кількість різних двійкових кодів довжини n дорівнює 2^n. Написати програму генерування всіх двійкових кодів довжини n.
3. Написати програму генерування двійкових кодів Грея порядку n.
4. Дана множина M={1, 2, …, m}. Кількість усіх підмножин множини M дорівнює 2^m. Написати програму генерування всіх підмножин множини M.
5. Дана множина M={1, 2, …, m}. Написати програму генерування всіх .k-елементних підмножин множини M, де k<m.
6. Дано натуральне число n. Записати його в системі числення з основою p, де p - просте число, наприклад 13.
7. Дано дві послідовності довжини n із цілих чисел. Вивести ті числа першої послідовності, що не містяться в другій послідовності.
8. Число перестановок n-елементної множини дорівнює n!. Написати програму генерування всіх перестановок множини M={1, 2, …, m}.

☝ Питання для самоконтролю

1. Що таке одновимірний масив?
2. Синтаксис оголошення масиву.
3. Чи можна змінювати розмірність або кількість елементів масиву під час виконання програми?
4. Як здійснюється доступ до елементів масиву?
5. Які дії над масивами можна виконувати в мові C?
6. Які дії можна виконувати над елементами масиву?
7. Що робить компілятор, коли зустрічає таке оголошення double a[6]?
8. Як розташовуються елементи масиву в пам’яті комп’ютера?
9. Чим масив відрізняється від файлу?
10. Які з перелічених оголошень правильні?
    a. int double mas1[10];
    b. unsigned int mas2[10];
    c. char mas3[0..10];
    d. int mas4[-10..10];
    e. int mas5[‘a’];
    f. char mas6[-5];
    g. int mas7[20-10+1];
    h. int mas8[20.5];
11. Чи можна ініціалізувати масив більшою кількістю значень, ніж вказано в його оголошенні?
12. Вкажіть правильні ініціалізації масиву. Які з них не мають сенсу?
    a. int mas[5]={};
    b. int mas[5]={2, -2};
    c. int mas[5]={2, -2, 0, 4, -1};
    d. int mas[5]={2, -2, 0, 4, -1, 5, 8};
    e. int mas[]={};
    f. int mas[]={2, -2, 0, 4, -1, 5, 8};
13. Є фрагмент програми:

```
double vec[10] = {2.4, -5.6, 4.4, 0.5, -2.2};
double *p;
p=vec+4;
```

Вкажіть значення кожного з наступних виразів:

1) vec[0] < vec[1]
2) *vec+2
3) *(vec+2)
4) *p*2
5) 2**p
6) *--p

Практичне заняття № 7

Тема: ***Багатовимірні масиви. Матриці***

Студент повинен знати: означення масиву, синтаксис оголошення масиву, ініціалізацію масиву, перегляд елементів масиву та виконання дій над ними.

Теоретичні відомості

Багатовимірний масив, як і одновимірний, є скінченною послідовністю однотипних елементів. У мові С багатовимірний масив трактується як масив масивів, тобто масив, елементами якого є масиви меншої вимірності (підмасиви). Найчастіше використовуються двовимірні масиви, які часто називають матрицями або таблицями. Багатовимірний масив оголошується за синтаксисом

<тип_елементів> <ім'я_масиву>[<кількість1>][<кількість2>]…[<кількістьN>];

До типу_елементів, імені_масиву та кількостей_елементів висуваються такі ж самі вимоги, що і для одновимірних масивів. Кожний елемент масиву має індекси (номери). Індексація кожної вимірності починається з нуля. Для масиву компілятор виділяє неперервну ділянку пам'яті.

Приклади оголошень масивів:

```
int matr[10][5];
double kub[6][6][6];
```

Перший масив двовимірний, другий – тривимірний. Для кожного із них виділяється неперервна ділянка пам'яті. Перший масив можна трактувати як матрицю із 10 рядків і 5 стовпців. Він розташовується в пам'яті рядками - спочатку перший рядок matr[0], потім другий - matr[1], і так далі, десятий - matr[9].

При означенні масиву відбувається його ініціалізація. Приклади означень:

```
int matr[4][3] = {1, 2, 3, -1, 5, -6, 7, 9,3, 0, -1, 4};
int matr[][3] = {1, 2, 3, -1, 5, -6, 7, 9,3, 0, -1, 4};
int matr[4][3] = {{1, 2, 3}, {-1, 5, -6}, {7, 9,3}, {0, -1, 4}};
int matr[4][3] = {{1, 2, 3}, {-1, 5}, {7}, {0}};
```

В означенні багатовимірного масиву дозволено опускати розмірність найстаршого виміру.

У мову С немає операцій над масивами, як єдиним цілим. Операції можна виконувати тільки над елементами масиву, а обсяг допустимих операцій залежить від типу елементів. Для доступу до елементів масиву використовують індекси або вказівники. Приклади звертань через індекси: matr[0][0] – перший елемент першого рядка, matr[3][2] – третій елемент четвертого рядка, matr[i][j] – (j+1)-й елемент (i+1)-го рядка. Елементи масиву є змінними, а тому їм можна присвоювати значення, загалом, використовувати у виразах.

Приклад заповнення масиву matr з клавіатури:

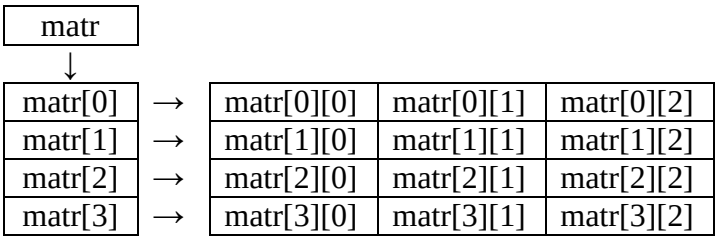
```
for (i=0; i<4; i++)
        for (j=0; i<3; j++)
                scanf("%d", &matr[i][j]);
```

Приклад виведення цього масиву:

```
for (i=0; i<4; i++)
        {   for (j=0; i<3; j++)
                printf("%d  ", matr[i][j]);
            printf("\n");
        }
```

Ім'я масиву є константним вказівником на масив і воно використовується для доступу до елементів масиву. Розглянемо це на прикладі масиву matr. Ім'я matr – це вказівник на початок масиву, який складається із чотирьох елементів: matr[0], matr[1], matr[2], matr[3]. Однак, кожний із цих елементів є масивом із трьох елементів. Тому matr[0] є вказівником на початок масиву із елементів matr[0][0], matr[0][1], matr[0][2].

Аналогічно, matr[1] є вказівником на початок масиву із елементів matr[1][0], matr[1][1], matr[1][2]. І так далі. Схематично це зображено на малюнку.



Таким чином, matr – це подвійний вказівник, а matr[0], matr[1], matr[2], matr[3] – звичайні вказівники (одинарні). Нижче у таблиці наведені приклади рівносильних записів.

Звертання через вказівники	Рівносильний запис
*matr	matr[0], &matr[0][0]
*(matr+2)	matr[2], &matr[2][0]
*(matr+i)	matr[i], &matr[i][0]
*(matr+i)+j	matr[i]+j, &matr[i][j]
((matr+i)+j)	matr[i][j]
**matr	matr[0][0]

У n-вимірному масиві ім’я масиву є n-арним вказівником.

Через вказівники можна ввести елемент масиву з клавіатури. Наприклад,

```
for (i=0; i<4; i++)
        for (j=0; i<3; j++)
                scanf(“%d”, *(matr+i)+j);
```

Аналогічно можна організувати виведення елементів масиву

```
for (i=0; i<4; i++)
        {   for (j=0; i<3; j++)
                printf(“%d   ”, *(*(matr+i)+j));
            printf(“\n”);
        }
```

Приклад 1

Дано двовимірний масив цілих чисел a[n,m]. Вивести цей масив на екран і окремо вивести мінімальний елемент даного масиву.

Аналіз задачі

Щоб знайти мінімальний елемент двовимірного масиву, необхідно змінній min присвоїти початкове значення - перший елемент масиву. Потім у циклі на кожному кроці порівнювати черговий елемент масиву з min, і якщо черговий елемент виявиться меншим, то змінній min присвоїти його значення. Переглянувши всі елементи масиву, знайдемо мінімальний.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 5                     // Кількість рядків у масиві
#define M 6                     // Кількість стовпців у масиві

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i, j, min;                  // min – змінна мінімального елемента масиву
        int a[N][M];                    // Оголошення масиву
        printf ("Введіть елементи масиву\n");
```

```
        for (i=0; i<N; i++)
                for (j=0; j<M; j++)
                        scanf("%d", &a[i][j]);
        printf ("Виведення елементів масиву\n");
        for (i=0; i<N; i++)
                {
                        for (j=0; j<M; j++)
                                printf("%5d", a[i][j]);
                        printf ("\n");
                }
        // Пошук мінімального елемента масиву
        min = a[0][0];                  // Початкове значення мінімального елемента
        for (i=0; i<N; i++)
                for (j=0; j<M; j++)
                        if (a[i][j]<min)
                                min = a[i][j];
        printf («Мінімальний елемент = %d\n», min);
        system(«pause»);
        return 0;
}
```

```
Введіть елементи масиву
   1   2   4  -5  -2  -1
  99  11  44  55  78  99
 -12  -9  -8  -7  -8  -6
 -11   9   7   8   7 -10
  23 -44  -6  -4  33  65
Виведення елементів масиву
    1    2    4   -5   -2   -1
   99   11   44   55   78   99
  -12   -9   -8   -7   -8   -6
  -11    9    7    8    7  -10
   23  -44   -6   -4   33   65
Мінімальний елемент = -44
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми

Приклад 2

Дано матриці A, B, C дійсних чисел розмірності nxm. Обчислити величину $\frac{\|A\|+\|B\|+\|C\|}{\|A+B+C\|}$, *де* $\|D\| = \max_j |D[1, j]| + \max_j |D[2, j]| + ... + \max_j |D[n, j]|$.

Аналіз задачі

Величина $\|D\|$ дорівнює сумі найбільших елементів рядків. Її у програмі потрібно обчислювати чотири рази. Тому для обчислення $\|D\|$ доцільно створити функцію, у яку через параметри передається матриця. Функція повинна повертати суму. Прототип такої функції матиме вигляд

```
double SumaMax(double d[][M], int n, int m);
```

Функція повертає результат типу double, її ім'я SumaMax, вона має три формальні параметри. Перший параметр double d[][M] служить для передачі двовимірного масиву. Така форма запису показує, що параметр d є вказівником на перший підмасив двовимірного масиву. Через параметри n і m передаються розмірності масиву. Код функції SumaMax наведений у програмі.

Прототип функції SumaMax також можна було б задати так

```
double SumaMax(double (*d)[M], int n, int m);
```

У цьому випадку явно вказано, що параметр d є вказівником на перший підмасив. Зауважимо, що форма запису параметра double (*d)[M] означає, що d є вказівником на одновимірний масив із M елементів типу double, тобто d може адресувати будь який рядок матриці. Код функції SumaMax для цього випадку можна залишити таким як у програмі, помінявши лише заголовок функції. Однак можна створити код, у якому звертання до елементів масиву буде здійснюватися через вказівники. Приклад такого коду:

```
double SumaMax(double (*d)[M], int n, int m)
{       double suma=0, max, *p;      // max – найбільший елемент рядка, suma – сума
                                                                  найбільших елементів
        int i, j;
        for (i=0; i<n; i++, d++)        // Цикл для руху по рядках, отримує приріст вказівник d
        {       max=**d;                // Початкове значення max i-го рядка
                for (j=1, p=*d+1; j<m; j++, p++)     // Вказівник p пробігає рядок
                        if (*p>max)
                                max=*p;
                suma+=max;              // Добавлення до суми max i-го рядка
        }
        return suma;
}
```

Оскільки ім’я двовимірного масиву є подвійним вказівником, то можна було б задати прототип функції у вигляді

```
double SumaMax(double **d, int n, int m);
```

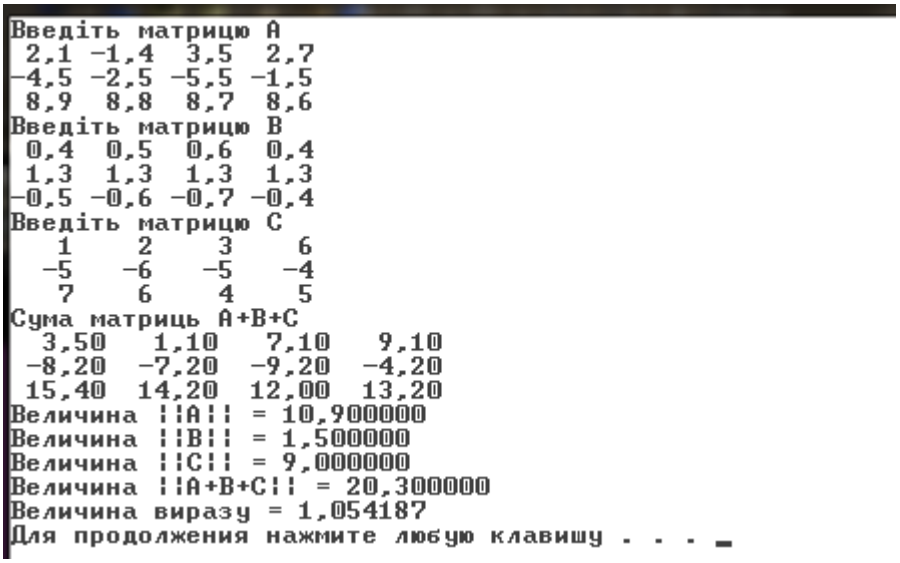
Тут параметр d є просто подвійним вказівником на тип double. Однак через такий параметр не можна передати двовимірний масив, бо невідома величина підмасивів масиву. Наприклад, в масиві a[10][5] підмасивами є рядки із 5-ти елементів.

Загальне правило передачі багатовиміри масивів у функцію можна сформулювати так. Якщо потрібно передати у функцію багатовимірний масив, наприклад int a[3][4][6], то формальний параметр може мати вигляд int c[][4][6] або int (*c)[4][6].

Далі наведений код програми

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 3                                     // Кількість рядків матриці
#define M 4                                     // Кількість стовпців матриці

double SumaMax(double d[][M], int n, int m);        // Прототип функції
int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i, j;
        double a[N][M];                         // Масив для матриці A
        double b[N][M];                         // Масив для матриці B
        double c[N][M];                         // Масив для матриці C
        double s[N][M];                         // Масив для суми матриць A+B+C
        double suma_a, suma_b, suma_c, suma_s;          // Оголошення змінних для величин
                                                                  ||A||, ||B||, ||C||, ||A + B + C|| відповідно
        double resultat;                        // Змінна для виразу (||A|| + ||B|| + ||C||) / ||A + B + C||

        printf ("Введіть матрицю A\n");
        for (i=0; i<N; i++)
                for (j=0; j<M; j++)
```

```
                scanf("%lf", &a[i][j]);
        printf ("Введіть матрицю B\n");
        for (i=0; i<N; i++)
                for (j=0; j<M; j++)
                        scanf("%lf", &b[i][j]);
        printf ("Введіть матрицю C\n");
        for (i=0; i<N; i++)
                for (j=0; j<M; j++)
                        scanf("%lf", &c[i][j]);
        // Обчислення A+B+C
        for (i=0; i<N; i++)
                for (j=0; j<M; j++)
                        s[i][j] = a[i][j]+b[i][j]+c[i][j];

        printf ("Сума матриць A+B+C\n");
        for (i=0; i<N; i++)
                {       for (j=0; j<M; j++)
                                printf("%6.2lf ", s[i][j]);
                        printf("\n");
                }
        suma_a=SumaMax(a, N, M);                        // Обчислення ||A||
        printf("Величина ||A|| = %lf\n", suma_a);
        suma_b=SumaMax(b, N, M);                        // Обчислення ||B||
        printf("Величина ||B|| = %lf\n", suma_b);
        suma_c=SumaMax(c, N, M);                        // Обчислення ||C||
        printf("Величина ||C|| = %lf\n", suma_c);
        suma_s=SumaMax(s, N, M);                        // Обчислення ||A + B + C||
        printf("Величина ||A+B+C|| = %lf\n", suma_s);
        resultat=(suma_a+suma_b+suma_c)/suma_s;
        printf("Величина виразу = %lf\n", resultat);

        system("pause");
        return 0;
}
double SumaMax(double d[][M], int n, int m)
{
        double suma=0, max;             // Змінна suma для величини ||D||,
                                         max для пошуку максимального елемента рядка
        int i, j;
                for (i=0; i<n; i++)
        {       max=d[i][0];            // Початкове значення max i-го рядка
                for (j=1; j<m; j++)
                        if (d[i][j]>max)
                                max=d[i][j];
                suma+=max;              // Добавлення до суми max i-го рядка
        }
        return suma;
}
```



Результат роботи програми

Приклад 3

Є таблиця T результатів деякого шахового турніру, у якому приймали участь **n** шахістів (**n**>2): T[i,j]=2, якщо i-й учасник виграв у j-го (при цьому T[j,i]=0); T[i,j]=1, якщо i-й та j-й учасники зіграли в нічию (також T[j,i]=1); T[i,i]=0. Вивести номери учасників змагання в порядку не зростання набраних ними очок.

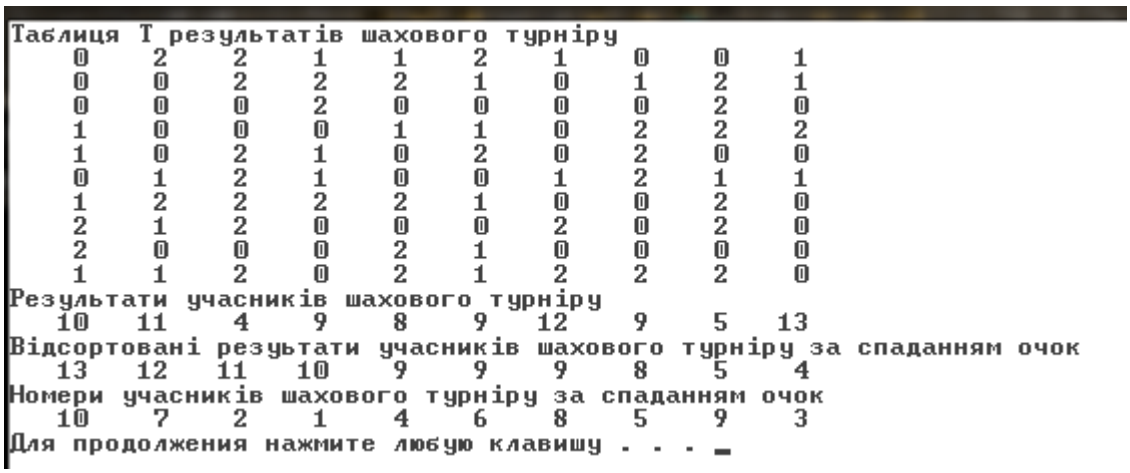
Аналіз задачі

Таблиця T результатів турніру є квадратною матрицею порядку **n**, яка у програмі задається двовимірним масивом. Спочатку потрібно заповнити цей масив. Досить задати довільним чином значення елементів масиву над головною діагоналлю, бо якщо задано значення T[i,j], то T[j,i] обчислюється з умови задачі. На головній діагоналі потрібно розмістити нулі.

Після заповнення матриці результатів потрібно обчислити для кожного учасника суму набраних ним очок. Ці суми будемо зберігати у одновимірному масиві sum[n] за кількістю учасників змагання. Також треба запам’ятати номери учасників, наприклад, у масиві num[n]. Далі потрібно упорядкувати масив sum[n] за не зростанням (спаданням) набраних учасниками змагання очок. У процесі сортування sum[n] паралельно аналогічні дії будемо виконувати над масивом num[n]. Після завершення сортування масив num[n] буде містити номери учасників змагання у потрібному порядку. Можна також паралельно сортувати матрицю T.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 10                    // Кількість учасників турніру

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i, j, k;
        int t[N][N];                    // Масив таблиці T результатів турніру
        int sum[N];                     // Масив для сум очок учасників турніру
        int num[N];                     // Масив для номерів учасників турніру
        // Заповнення матриці T шахового турніру
        for (i=0; i<N; i++)
                for (j=i+1; j<N; j++)
                {       t[i][j]=rand()%3;       // Значення елементів над головною діагоналлю
```

```
                        // Значення елементів під головною діагоналлю
                        if (t[i][j]==2)   t[j][i]=0;
                        if (t[i][j]==1)   t[j][i]=1;
                        if (t[i][j]==0)   t[j][i]=2;
                }
// Значення елементів на головній діагоналі – всі нулі
        for (i=0; i<N; i++)
                t[i][i]=0;
        printf ("Таблиця Т результатів шахового турніру\n");
        for (i=0; i<N; i++)
                {       for (j=0; j<N; j++)
                                printf ("%5d", t[i][j]);
                        printf ("\n");
                }
        // Номери учасників турніру
        for (i=0; i<N; i++)
                num[i]=i+1;
        // Обчислення сум очок учасників
        for (i=0; i<N; i++)
                {       sum[i]=0;
                        for (j=0; j<N; j++)
                                sum[i]+=t[i][j];
                }
        printf ("Сумарні результати учасників шахового турніру\n");
        for (i=0; i<N; i++)
                        printf ("%5d", sum[i]);
        printf ("\n");
        // Сортування результатів за спаданням методом бульбашки
        for (i=N-2; i>=0; i--)
                for (j=0; j<=i; j++)
                        if (sum[j]<sum[j+1])
                        {       // Перестановка елементів масиву sum
                                k=sum[j];
                                sum[j]=sum[j+1];
                                sum[j+1]=k;
                                // Перестановка елементів масиву num
                                k=num[j];
                                num[j]=num[j+1];
                                num[j+1]=k;
                        }
        printf ("Відсортовані резуьтати учасників шахового турніру за спаданням очок\n");
        for (i=0; i<N; i++)
                printf ("%5d", sum[i]);
        printf ("\n");
        printf ("Номери учасників шахового турніру за спаданням очок\n");
        for (i=0; i<N; i++)
                printf ("%5d", num[i]);
        printf ("\n");
        system("pause");
        return 0;
}
```



Результат роботи програми.

Задачі

1. Дано матрицю A[m,n] дійсних чисел. Обчислити суму елементів для кожного рядка матриці.
2. Здійснити обхід матриці по спіралі за годинниковою стрілкою, починаючи від її лівого верхнього кута. Вивести елементи матриці у порядку обходу.
3. Колода з 52 гральних карт може бути представлена у вигляді двовимірного масиву розмірності 4x12.

	Туз	Двійка	Трійка	…	Дев'ятка	Десятка	Валет	Дама	Король
Черви									
Бубни									
Трефи									
Піки									

Перед грою колода карт тасується і тим самим визначається порядок здавання карт. Скласти програму, яка випадковим чином заповнює таблицю числами від 1 до 52. Вказати ефективний алгоритм тасування. Після того, як колода потасована, треба роздати карти. Скласти програму роздавання карт. Придумати ефективний алгоритм роздавання.

☺ Індивідуальні завдання

Основний рівень

1. Дано масив A[m,n] цілих чисел. Вивести цей масив на екран, підрахувати кількість від'ємних елементів та вивести їх індекси.
2. Дано масив A[m,n] дійсних чисел та число **a**. Вивести цей масив на екран, підрахувати кількість елементів, рівних **a**, та вивести їх індекси.
3. Дано масив A[m,n] цілих чисел. Вивести цей масив на екран, підрахувати кількість елементів, які більші за –5, але менші за 5, та вивести їх індекси.
4. Дано масив A[m,n] дійсних чисел. Вивести цей масив на екран, підрахувати кількість додатних елементів та вивести їх індекси.
5. Дано масив A[m,n] цілих чисел. Вивести цей масив на екран, підрахувати кількість нульових елементів та вивести їх індекси.
6. Дано масив A[m,n] дійсних чисел. Вивести цей масив на екран, підрахувати кількість найменших елементів та вивести їх індекси.

7. Дано масив A[m,n] дійсних чисел. Вивести цей масив на екран, підрахувати кількість найбільших елементів та вивести їх індекси.
8. Дано масив A[m,n] дійсних чисел. Вивести цей масив на екран, знайти у першому стовпці найбільший елемент, а в останньому – найменший. Вивести ці елементи на екран.
9. Дано масив A[m,n] цілих чисел. Вивести цей масив на екран і підрахувати кількості нульових, додатних і від'ємних елементів у останньому стовпці.
10. Дано масив A[m,n] із 0 та 1. Вивести цей масив на екран, підрахувати кількості 0 та 1.
11. Дано масив A[m,n] цілих чисел від 0 до 100. Вивести на екран цей масив і рядок, у якому вперше зустрілося число 50. Якщо такого числа немає, то поінформувати про це.
12. Дано масив A[m,n] цілих чисел. Вивести на екран цей масив і рядок, у якому знаходиться найбільший елемент (він єдиний).
13. Дано масив A[m,n] із 0 та 1. Вивести цей масив на екран і підрахувати кількість нульових рядків.
14. Дано масив A[m,n] цілих чисел. Вивести цей масив на екран і вивести окремо рядки, у яких на останньому місці знаходиться найменший елемент рядка.
15. Дано масив A[m,n]цілих чисел. Вивести цей масив на екран і вивести окремо стовпчики, у яких на першому місці знаходиться найбільший елемент стовпчика.

Підвищений рівень

А

1. Дано масив A[m, n] цілих чисел. Вивести його на екран, знайти мінімальний елемент у кожному стовпчику і замінити його на 0. Вивести новий масив на екран.
2. Дано масив A[m, n] цілих чисел. Вивести його на екран, знайти максимальний елемент у кожному рядку і замінити його на 1. Вивести новий масив на екран.
3. Дано матрицю A[m,m] дійсних чисел. Вивести її на екран, поміняти місцями дві діагоналі матриці. Вивести нову матрицю.
4. Дано матрицю A[m,m] цілих чисел. Вивести її на екран і визначити, чи є вона симетричною відносно головної діагоналі.
5. Дано матрицю A[m,m] дійсних чисел. Вивести її на екран і знайти суму всіх елементів, які не знаходяться на жодній з двох діагоналей .
6. Дано матрицю A[m,m] цілих чисел. Вивести її на екран і обчислити окремо суми елементів над та під головною діагоналлю. Яка з цих сум більша?
7. Дано матрицю A[m,n] дійсних чисел. Вивести цю матрицю на екран, знайти матрицю, транспоновану до даної, і вивести її на екран.
8. Дано матрицю A[m,n] дійсних чисел. Визначити кількість "особливих" елементів матриці та вивести їх індекси. Вважати елемент "особливим", якщо він більший за суму елементів останнього стовпця.
9. Дано матрицю A[m,m] цілих чисел. Визначити кількість "особливих" елементів у рядках матриці та вивести їх індекси. Вважати елемент рядка "особливим", якщо зліва від нього знаходиться менший елемент, а справа більший.
10. Дано матрицю A[m,m] цілих чисел. Визначити, чи буде дана матриця магічним квадратом, тобто такою, у якої суми елементів у всіх рядках і стовпцях однакові.
11. Дано матрицю A[m,m] цілих чисел. Вивести цю матрицю на екран і визначити, чи є вона симетричною відносно неголовної діагоналі.
12. Дано матрицю A[m, n] цілих чисел. Вивести цю матрицю на екран та знайти максимальний і мінімальний елементи матриці. Перетворити дану матрицю, замінивши всі елементи, що більші мінімального та менші максимального, на 0. Вивести нову матрицю.

13. Дано матрицю A[m,n] дійсних чисел. Вивести цю матрицю на екран, замінити елементи головної діагоналі на середні гармонійні елементи відповідного рядка (середнє гармонійне - це сума елементів, обернених до даних, поділена на їх кількість).
14. Дано два масиви A[n,m], B[m,k] цілих чисел. Вивести обидві матриці на екран, обчислити їх добуток і результат вивести на екран.
15. Дано матрицю A[m,m] із 0 та 1. Вивести цю матрицю на екран і підрахувати, скільки в ній нульових рядків та стовпців. Вивести номери цих рядків і стовпців.

Б

1. Таблиця кола футбольного чемпіонату задана квадратною матрицею порядку **n**. Знайти число команд, що мають більше перемог, ніж поразок.
2. Таблиця кола футбольного чемпіонату задана квадратною матрицею порядку **n**. Визначити номери команд, що пройшли чемпіонат без поразок.
3. Таблиця кола футбольного чемпіонату задана квадратною матрицею порядку **n**. З'ясувати, чи є хоч одна команда, що виграла більше половини ігор.
4. Таблиця кола футбольного чемпіонату задана квадратною матрицею порядку **n**. Вивести номери команд, що посіли відповідно перше, друге та третє місця.
5. Таблиця кола футбольного чемпіонату задана квадратною матрицею порядку **n**. Вивести номери команд, що посіли два останніх місця.
6. Для заданої цілочислової матриці A[m,n] надрукувати індекси тих її елементів, які є найменшими у своєму рядку і одночасно найбільшими у своєму стовпці.
7. Дана матриця A[n,m] дійсних чисел. Упорядкувати її рядки за не спаданням сум елементів рядків.
8. Дана матриця A[n,m] дійсних чисел. Упорядкувати її стовпці за спаданням сум елементів стовпців.
9. Дана матриця A[n,m] дійсних чисел. Упорядкувати її рядки за не спаданням найменших елементів рядків.
10. Дана матриця A[n,m] дійсних чисел. Упорядкувати її рядки за не спаданням найбільших елементів рядків.
11. Шахову дошку можна представити символьною матрицею 8x8. Дано натуральні числа p і q ($1 \le p \le 8$, $1 \le q \le 8$), що визначають місцезнаходження ферзя. Відповідно в цю клітинку матриці записати символ Ф. Поля, що знаходяться під загрозою ферзя, заповнити символом *, а решту полів – символом 0. Вивести заповнену матрицю.
12. Дана квадратна матриця A[m,m] цілих чисел. Обчислити суму елементів у зафарбованих областях.

а) б) в) г)

♖ Додаткові задачі

1. Елемент матриці назвемо сідловою точкою, якщо він є найменшим у своєму рядку і одночасно найбільшим у своєму стовпці або, навпаки, є найбільшим у своєму рядку і одночасно найменшим у своєму стовпці. Для заданої цілочислової матриці A[m,n] надрукувати індекси всіх її сідлових точок.
2. Гра «Життя» відбувається на прямокутному полі комірок, кожна із яких може містити організм. Комірка має 8 сусідів. «Зайнято(К)» означає кількість комірок, сусідніх із коміркою К і вже заповнених організмами. Конфігурація нового покоління організмів утворюється з попереднього покоління за правилами:
   a. організм у комірці К переходить до наступного покоління, якщо виконується умова 2≤зайнято(К)≤3, інакше помирає;
   b. організм народжується в порожній комірці К, якщо зайнято(К)=3.

   Написати програму, яка задає початкову конфігурацію і читає число **n**, обчислює **n**-е покоління у відповідності з правилами та виводить знайдену таблицю.
5. Задати матрицю **mxn** (**m, n** ≥ 3). Починаючи з лівого нижнього кута матриці та рухаючись лише праворуч і догори, досягти її правого верхнього кута і вибрати при цьому такі значення елементів, що їх сума буде максимальною. Вивести перелік вибраних елементів.
6. Дано систему **n** лінійних рівнянь з **n** невідомими

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i \qquad (i=1,2,\ldots,n)$$

   Вважаючи, що визначник |A|≠0, написати програму розв'язання системи методом Гауса.
7. Дано квадратну матрицю A **n**-го порядку. Знайти обернену матрицю A^{-1}, якщо вона існує, або встановити, що оберненої не існує. Написати відповідну програму.
8. Дано квадратну матрицю A **n**-го порядку. Створити програму обчислення визначника матриці.

☝ Питання для самоконтролю

1. Що таке багатовимірний масив?
2. Синтаксис оголошення багатовимірного масиву.
3. Чи можна змінювати розмірність або кількість елементів масиву під час виконання програми?
4. Які ви знаєте способи оголошення масивів?
5. Які ви знаєте дії над масивами?
6. Які дії можна виконувати над елементами масиву?
7. Як розташовані елементи багатовимірного масиву в пам'яті комп'ютера?
8. Як здійснюється доступ до елемента масиву?
9. Які з перелічених оголошень правильні?
   a. int Mas[1..10][1..20];
   b. int Mas[10][20];
   c. int Mas[10][];
   d. int Mas[][20];
   e. int Mas[][];
   f. int Mas[1.5][2.5];
   g. int Mas['x']['y'];
   h. int Mas[0][0];
10. У програмі оголошений масив:
    int Mas[][6]={{5, 12, -7}, {-3, 6, 3, 8), {14, 4, -6, 2}};
    Вкажіть значення кожного з наступних виразів:
    a. Mas[2][0]
    b. *(Mas[1]+2)
    c. **Mas
    d. *Mas[1]
    e. (*Mas)[1]
    f. *(*(Mas+1)+3)

Практичне заняття № 8

Тема: ***Рядки***

Студент повинен знати: означення рядка, синтаксис оголошення рядків, ініціалізацію рядків, стандартні функції для обробки рядків.

Теоретичні відомості

Рядок – це скінченна послідовність символів кодової таблиці комп'ютера. У мові С рядки є одновимірними масивами, елементи яких мають тип char. Останнім символом рядка повинен бути так званий нуль-символ ('\0'), код якого дорівнює 0. З кожним рядком пов'язується вказівник на початок даного рядка. У всьому іншому рядки повністю зберігають властивості масивів.

Синтаксис оголошення рядка, який є змінною, такий

char <ім'я_рядка> [<кількість_символів>];

При означенні рядків відбувається їхня ініціалізація. Наприклад,

```
char s1[10] = {'m', 'a', 'm', 'a', '\0'};
char s2[10] = "mama";
char s3[] = "mama";
```

Значенням усіх трьох рядків є слово mama. Однак,

```
char s4[10] = {'m', 'a', 'm', 'a'};
```

s4 не є рядком. З ним можна працювати лише як з масивом.

Процеси опрацювання рядків у С-програмах базуються на двох основних властивостях:

1. ім'я рядка є константним вказівником на його перший символ;
2. кінець рядка задається нуль-символом '\0'.

При звертанні до символів рядка можна використовувати індексну або вказівникові форми.

Бібліотека мови С містить функції для введення/виведення символів та рядків. Прототипи цих функцій оголошені в файлі stdio.h.

Для введення окремого символу з клавіатури (функція читає символ зі стандартного потоку stdin) можна використовувати функцію getchar, прототип якої має вигляд:

int getchar(void);

Функція getchar не має параметрів і повертає код зчитаного символу, доповненого до типу int старшим нульовим байтом. У разі виявлення помилки введення функція повертає макроконстанту EOF. Для більшості систем значення цієї макроконстанти дорівнює -1. Приклад використання функції

```
char symb;
symb = getchar();
```

Виведення одного символу виконує функція

int putchar(int sym);

Через параметр sym у функцію передається символ (код символу), який відобразиться на екрані (насправді символ записується у стандартний потік виведення stdout). При успішному виконанні функція повертає код символу, а при невдачі – макроконстанту EOF. Приклад виведення

putchar('a');

Для форматного введення рядків використовується функція scanf, а для виведення – функція printf. Однак scanf читає рядок до першого пробільного символу. Тому її доцільно використовувати для введення слів.

Для введення довільних рядків використовують функцію gets, прототип якої має вигляд

char *gets(char *st);

Ця функція читає рядок, введений з клавіатури, і записує його в масив, адреса початку якого задається параметром-вказівником st. Ознакою кінця введення слугує символ нового рядка ‘\n’, який заноситься в буфер введення при натисненні клавіші Enter. Замість символу ‘\n’ функція записує у рядок нуль-символ ‘\0’.

У разі успішного завершення gets повертає вказівник на початок введеного рядка, тобто адресу &st[0] першого символу рядка, а в разі виникнення помилки – порожній вказівник NULL.

Виведення рядка здійснює функція

int puts(const char *st);

Параметр st - це вказівник на рядок, який виводиться. При виведенні функція puts замінює символ ‘\0’ на ‘\n’, тобто після виведення рядка курсор перейде на новий рядок.

Далі наводимо перелік основних функцій для роботи з рядками і символами.

Основні функції класифікації та зміни символів (файл ctype.h)

Функція	**Призначення**
класифікації:	Перевіряє, чи символ sym є:
int **isalpha**(sym)	малою або великою латинською літерою
int **isdigit**(sym)	десятковою цифрою
int **isalnum**(sym)	латинською літерою або десятковою цифрою
int **isxdigit**(sym)	шістнадцятковою цифрою
int **isspace**(sym)	пробільним символом (символом пробілу, нового рядка, горизонтальної чи вертикальної табуляції)
int **islower**(sym)	малою латинською літерою
int **isupper**(sym)	великою латинською літерою
перетворення	Повертає
int **tolower**(sym)	малу латинську літеру, якщо sym є велика латинська літера
int **toupper**(sym)	велику латинську літеру, якщо sym є мала латинська літера

Основні функції для роботи з рядками (файл string.h)

Прототип функції	**Призначення**
char * **strcpy**(char *st, const char *s)	Копіює рядок s (з ‘\0’ включно) у рядок, що адресується вказівником st. Функція повертає st – адресу скопійованого рядка.
char * **strncpy**(char *st, const char *s, int n)	Копіює не більше ніж n перших символів рядка s у рядок, що адресується вказівником st. Якщо довжина рядка s менша за n, то буде скопійований весь рядок s. Якщо скопійована група символів не закінчується ‘\0’, то нуль-символ у st не заноситься. Функція повертає st – адресу скопійованого рядка.
char * **strcat**(char *st, const char *s)	Виконує конкатенацію рядка st з s, тобто до кінця рядка st добавляється рядок s і утворюється рядок із двох рядків. У рядку st повинно буди достатньо місця, щоб добавився рядок s. Функція повертає значення st – адресу доповненого рядка.
char * **strncat**(char *st, const char *s, int n)	Аналог **strcat**(), але долучає до st тільки n початкових символів з s; у кінець об’єднаного рядка заноситься ‘\0’.
int **strcmp**(const char *s1, const char *s2)	Порівнює два рядки s1 і s2 (рядки порівнюються посимвольно так як у словнику). Повертає: від’ємне ціле число, якщо s1<s2; 0, якщо s1==s2; додатне ціле число, якщо s1>s2. При порівнянні розрізняються символи верхнього і нижнього регістрів.
int **strncmp**(const char *s1, const char *s2, int n)	Аналог **strcmp**(), але порівнює тільки n початкових символів рядків s1 і s2. Якщо якийсь із рядків

	короткий за n, то порівняння припиняється з досягненням ‘\0’.
int **stricmp**(const char *s1, const char *s2)	Аналог **strcmp**(), але при порівнянні не розрізняються символи верхнього і нижнього регістрів.
int **strnicmp**(const char *s1, const char *s2, int n)	Аналог **strncmp**(), але при порівнянні не розрізняються символи верхнього і нижнього регістрів.
unsigned **strlen**(const char *s)	Повертає довжину рядка s у символах (‘\0’ не враховується)
char * **strchr**(const char *s, int sym)	Шукає в рядку s символ sym. Повертає вказівник на перше входження символу sym у рядок s або NULL, якщо sym не зустрічається у рядку.
char * **strrchr**(const char *s, int sym)	Аналог **strchr**(). Повертає вказівник на останнє входження символу sym у рядок s або NULL, якщо sym не зустрічається у рядку.
char * **strstr**(const char *s1, const char *s2)	Шукає в рядку s1 підрядок s2. Повертає вказівник на початок першого входження рядка s2 у рядок s1 або NULL, якщо s2 не зустрічається у рядку.
char * **strtok**(char *st, const char *s)	Виділяє в рядку st лексеми, обмежені символами із рядка s. Повертає вказівник на знайдену лексему або NULL. Функцію **strtok**() можна використовувати для розбиття рядка st на підрядки (речення на слова). У рядку s записують усі символи, які є розділювачами в st, тобто розділяють st на лексеми.
char * **strdup**(const char *s)	Копіює рядок s (з ‘\0’ включно) в динамічну пам’ять, попередньо виділивши там ділянку потрібної довжини. Повертає адресу рядка в динамічній пам’яті.

Функції перетворення рядків у числа та чисел у рядки (файл stdlib.h)

Прототип функції	Призначення
int **atoi**(const char *s)	Виділяє в рядку s перше ціле десяткове число і перетворює його в дане типу int. Числу може передувати довільна кількість символів пробілу. Кінцем рядка вважається перший символ, що не належить до цифр. Повертає знайдене число у разі успішного перетворення, а в разі помилки – результат не визначений.
long **atol**(const char *s)	Аналог **atoi**(), але перетворює рядок у число типу long.
double **atof**(const char *s)	Аналог **atoi**(), але перетворює рядок у число типу double.
char * **itoa**(int n, char *st, int baze)	Перетворює ціле число n в рядок, що записується за адресою st. Параметр baze ($2 \le baze \le 36$) є основою системи числення, у якій записане число n. Функція записує від’ємне число зі знаком – тільки тоді, коли baze=10.
char * **ltoa**(long n, char *st, int baze)	Перетворює довге ціле число n в рядок, що записується за адресою st. Параметр baze ($2 \le baze \le 36$) є основою системи числення, у якій записане число n. Функція записує від’ємне число зі знаком – тільки тоді, коли baze=10.

char * **ultoa**(unsigned long n, char *st, int baze)	Перетворює беззнакове довге ціле число n в рядок, що записується за адресою st. Параметр baze ($2 \leq$ baze ≤ 36) є основою системи числення, у якій записане число n.

Приклад 1

Дано рядок. Підрахувати в ньому кількість «пробілів».

Аналіз задачі

Алгоритм розв'язання задачі очевидний. Потрібно посимвольно переглядати рядок та перевіряти черговий символ, чи є він пробілом.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"                    // Файл для роботи з рядками

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i, n;                      // n – кількість пробілів
        char s[128];                   // оголошення рядка
        printf ("Введіть рядок\n");
        gets(s);                       // Введення рядка з клавіатури
        n=0;
        for (i=0; i<strlen(s); i++)    // перегляд рядка
                if (s[i]==' ')     n++;
        printf ("Кількість пробілів у рядку = %d\n", n);
        system("pause");
        return 0;
}
```

```
Введіть рядок
На нас падав дощ.
Кількість пробілів у рядку = 3
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 2

Написати програму, яка переводить римські числа в десяткові. Для запису римських чисел використовуються лише цифри I, V, X.

Аналіз задачі

Для запису римських чисел використовуються цифри: I – 1, V – 5, X – 10, L – 50, C – 100, D – 500, M – 1000. Натуральні числа утворюються з римських цифр за принципом додавання або віднімання. Залежно від натурального числа комбінують додавання з відніманням, щоб уникнути чотирикратного повторення одної цифри. Цифра може повторюватися підряд двічі або тричі. За принципом додавання вищу цифру пишуть перед нижчою (наприклад, VII=5+1+1=7, XXIII=10+10+1+1+1=23, MDLXV=1000+500+50+10+5 = 1565), а за принципом віднімання перед вищою цифрою пишуть нижчу і її значення віднімають (наприклад, IV=5-1=4, IX=10-1=9, XL=50-10=40). Число MCMXLIII=1000+ 1000-100+50-10+1+1+1=1943 утворене комбінуванням додавання і віднімання.

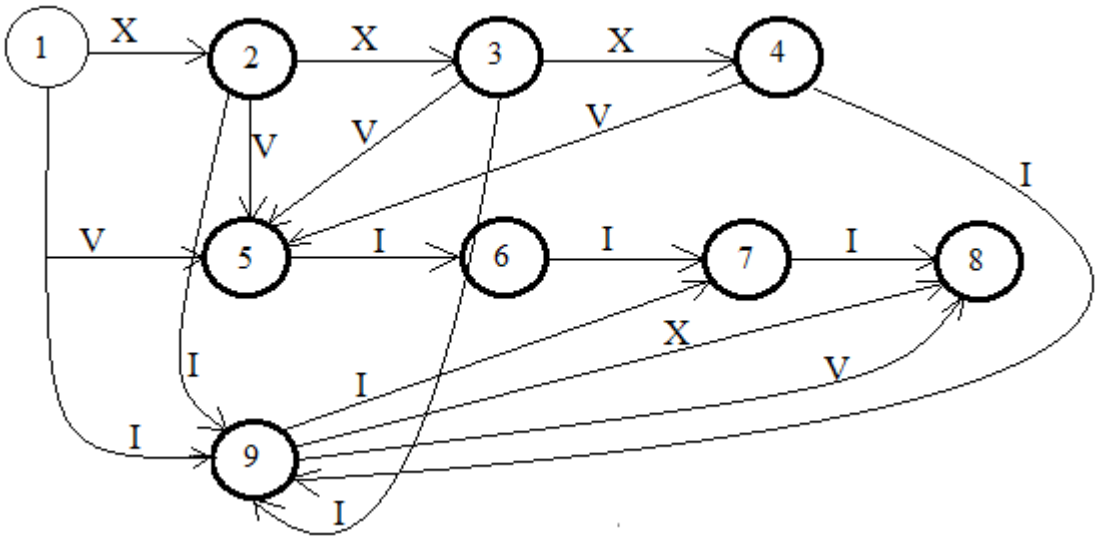
За умовою задачі ми обмежились лише числами, в записі яких використовуються лише цифри I, V, X, з метою створення невеликої програми. Найбільше число, яке може

бути утворене цими цифрами, дорівнює XXXIX=39. Загалом, можна створити програму, яка буде розпізнавати і переводити в десяткове число будь-яке римське число.

На вхід програми подається римське число, яке є послідовністю символів, тобто воно є рядком. Програма повинна розпізнавати правильність запису римського числа і якщо запис правильний, то виводити десятковий запис цього числа.

Задача розпізнавання рядків тексту є дуже важливою в програмуванні. Вона є центральною при побудові компіляторів мов програмування. У теорії компіляторів використовуються різноманітні алгоритми аналізу тексту. Багато із них ґрунтуються на побудові відповідного скінченного автомату. При розв'язанні даної задачі ми теж побудуємо скінченний автомат. Його можна зобразити у вигляді орієнтованого графа або задати у вигляді так званої таблиці переходів. Перший спосіб наочний і придатний для зображення невеликого автомата.

Скінченний автомат для розв'язання нашої задачі зображений на мал. 8.1.



Мал. 8.1. Детермінований скінченний автомат

Автомат має 9 станів. Вони зображені кружечками і занумеровані числами від 1 до 9. Стан 1 є початковим. У цьому стані автомат починає роботу. Решта станів є кінцевими. У них автомат може закінчувати роботу або продовжувати працювати далі. На вхід автомата буде подаватися рядок, який є вірною або невірною формою запису римського числа.

Стрілками зображені переходи автомата із одного стану в інший. Стрілки помічені римськими цифрами. Оскільки з вхідного рядка послідовно читатимуться римські цифри, то стан автомата і прочитана вхідна цифра визначатимуть наступний стан автомата.

Розглянемо роботу автомата на прикладах. Спочатку він перебуває у стані 1. Нехай з вхідного рядка прочитана цифра V. Оскільки із стану 1 виходить стрілка з поміткою V, то автомат перейде в стан 5. Якщо із вхідного рядка знову буде прочитана цифра V, а із стану 5 відповідної стрілки немає, то автомат припиняє роботу і повідомляє про помилку. Якщо автомат перебуває в стані 5, то він може перейти в стан 6 тільки тоді, коли прочитаний вхідний символ є цифрою I. Інших переходів із стану 5 немає.

Нехай на вхід автомата подається рядок XXV, який є правильним записом римського числа. Спочатку із рядка читається цифра X. Автомат із стану 1 переходить в стан 2. Потім читається знову цифра X. Автомат переходить в стан 3. Нарешті під дією символу V автомат перейде в стан 5. Читання вхідного рядка завершено – автомат перебуває в стані 5. Оскільки цей стан кінцевий, то автомат завершує роботу і видає інформацію про успішне розпізнавання вхідного рядка. Якби на вхід автомата був поданий рядок XXIIV, то автомат здійснив би ланцюжок переходів по станах 1→2→3→9→7 під дією символів XXII. Далі автомат перебуває в стані 7 і читається символ V. Автомат видає інформацію про помилку, бо немає такого переходу із стану 7. Отже число XXIIV не є правильним

римським числом. У його записі відбувається конфлікт між правилами додавання і віднімання.

Автомат на мал. 8.1 побудований за такими правилами:

- цифри I і X можуть зустрічатися підряд не більше трьох разів;
- цифра V не може повторюватися;
- після цифри X можуть знаходитися щонайбільше три цифри I або одна цифра V;
- після цифри V можуть знаходитися щонайбільше три цифри I;
- перед цифрою X може розташовуватися тільки одна цифра I або одна цифра V;
- перед цифрою V може розташовуватися тільки одна цифра I.

Зобразимо тепер автомат таблицею переходів (таблиця 8.1)

Таблиця 8.1. Таблиця автомата.

Стани	Вхідні символи		
	X	**V**	**I**
1	стан=2 (n=10)	стан=5 (n=5)	стан=9 (n=1)
2	стан=3 (n+=10)	стан=5 (n+=5)	стан=9 (n++)
3	стан=4 (n+=10)	стан=5 (n+=5)	стан=9 (n++)
4	помилка	стан=5 (n+=5)	стан=9 (n++)
5	помилка	помилка	стан=6 (n++)
6	помилка	помилка	стан=7 (n++)
7	помилка	помилка	стан=8 (n++)
8	помилка	помилка	помилка
9	стан=8 (n+=8)	стан=8 (n+=3)	стан=7 (n++)

Стовпці таблиці позначені вхідними символами, які використовуються для запису римського числа, а рядки позначені станами автомата. На перетині стовпців і рядків у клітинках записані стани, у які переходить автомат. Також у клітинках записане значення змінної n, яка призначена для збереження десяткового значення римського числа.

Пояснимо роботу автомата за таблицею. На початку він перебуває в стані 1. Якщо з вхідного рядка прочитана цифра X, то автомат переходить в стан 2, бо такий стан записаний у відповідній клітинці. Паралельно визначається значення n=10. Якщо з вхідного рядка читається наступна цифра, наприклад I, то автомат зі стану 2 переходить в стан 9. Значення n збільшиться на 1. Продовжуючи роботу автомата далі, ми або отримаємо десятковий запис римського числа (значення n) , або автомат видасть повідомлення про помилку. Останнє означає, що римське число записане невірно.

Алгоритм розв’язання задачі полягає в реалізації роботи автомата за таблицею 8.1. У програмі використаємо змінні: stan – для збереження стану автомата, n – для десяткового значення римського числа, flag – для перевірки правильності запису римського числа.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i;
        char rydok[10];         // Змінна для рядка, у якому зберігатиметься римське число
        int stan=1;             // Початковий стан автомата
        int flag=1;             // Початкове значення прапорця
        int n=0, m;
        printf ("Введіть римське число\n");
        gets(rydok);
        m=strlen(rydok);                // Довжина рядка
```

```
// Аналіз введеного з клавіатури римського числа
for (i=0; i<m; i++)
{       switch (stan)           // Реалізація таблиці переходів автомата
                {case 1: {if (rydok[i]=='X')
                                { stan=2; n=10; }
                                else
                                        if (rydok[i]=='V')
                                                { stan=5; n=5; }
                                                else
                                                        if (rydok[i]=='I')
                                                                { stan=9; n=1; }
                                                                else flag=0;
                                 break;
                                }
                 case 2: {if (rydok[i]=='X')
                                { stan=3; n+=10; }
                                else
                                        if (rydok[i]=='V')
                                                { stan=5; n+=5; }
                                                else
                                                        if (rydok[i]=='I')
                                                                { stan=9; n++; }
                                                                else flag=0;
                                  break;
                                }
                 case 3: {if (rydok[i]=='X')
                                { stan=4; n+=10; }
                                else
                                        if (rydok[i]=='V')
                                                { stan=5; n+=5; }
                                                else
                                                        if (rydok[i]=='I')
                                                                { stan=9; n++; }
                                                                else flag=0;
                                  break;
                                }
                 case 4: {if (rydok[i]=='V')
                                { stan=5; n+=5; }
                                else
                                        if (rydok[i]=='I')
                                                { stan=9; n++; }
                                                else flag=0;
                                  break;
                                }
                 case 5: {if (rydok[i]=='I')
                                { stan=6; n++; }
                                else flag=0;
                                 break;
                                }
                 case 6: {if (rydok[i]=='I')
                                { stan=7; n++; }
                                else flag=0;
```

```
                        break;
                      }
                case 7: {if (rydok[i]=='I')
                            { stan=8; n++; }
                            else flag=0;
                         break;
                        }
                case 8: {       flag=0;
                            break;
                        }
                case 9: {if (rydok[i]=='X')
                            { stan=8; n+=8; }
                            else
                                if (rydok[i]=='V')
                                    { stan=8; n+=3; }
                                    else
                                        if (rydok[i]=='I')
                                            { stan=7; n++; }
                                            else flag=0;
                         break;
                        }
                }
        if (flag==0) break;
    }
    if (flag)
        printf ("Десяткове значення римського числа = %d\n", n);
    else
        printf ("Невірний запис римського числа\n");
    system("pause");
    return 0;
}
```

```
Введіть римське число
XVIII
Десяткове значення римського числа = 18
Для продолжения нажмите любую клавишу . . .
```

```
Введіть римське число
XXIX
Десяткове значення римського числа = 29
Для продолжения нажмите любую клавишу . . .
```

```
Введіть римське число
VIX
Невірний запис римського числа
Для продолжения нажмите любую клавишу . . .
```

```
Введіть римське число
XXIIIVI
Невірний запис римського числа
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 3

Дано рядок слів, у якому між словами може стояти пробіл або розділові знаки – крапка, кома, двокрапка, крапка з комою, знак оклику, знак питання. Максимальна довжина слова у рядку може становити 10 символів. Вивести слова в алфавітному порядку, з яких складається рядок. Підрахувати кількість слів, які містять підслово «ах». У таких словах перше входження «ах» замінити на «ого». Вивести перетворені слова.

Аналіз задачі

З умови задачі випливає, що рядок складається зі слів української мови. Алгоритм розв'язання задачі складається з наступних кроків.

1. **Знайти слова, із яких складається рядок**. Для цього скористаємося функцією

   char * **strtok**(char *st, const char *s).

   Вона дозволяє знаходити у рядку лексеми (слова). Для збереження слів використаємо масив char slova[N][11], де N – кількість слів. Оскільки максимальна довжина слова 10 символів, то друга розмірність масиву дорівнює 11 (рядок обов'язково закінчується нуль-символом '\0'). Величина N невідома, але її потрібно обрати достатньою для кількості слів.

   Розглянемо використання функції **strtok** на прикладі рядка

   Направо! Кроком руш!

   У цьому рядку слова розділені пробілами та знаком «!». Тому значенням параметра s функції **strtok** є слово " !" із цих двох розділяючих символів, а параметр st адресує весь рядок.

   Організуємо виклик функції на прикладі фрагменту програми.

   ```
   char *s1;
   char s2[100]="Направо! Кроком руш!";
   s1=strtok(s2, " !");
   ```

   Функція **strtok** спочатку шукає у рядку s2 перший символ, відмінний від розділяючих символів (якщо зустрічаються розділяючи символи, то вони ігноруються). Для даного рядка перший не розділяючий символ – це буква «Н». Вона є початковим символом шуканого слова. Далі функція шукає перший розділяючий символ. Буде знайдений знак «!», який замінюється на нуль-символ, тобто рядок матиме вигляд

   Направо\0 Кроком руш!

   Цей нуль-символ є кінцем виділеного слова. Функція **strtok** повертає вказівник на знайдене слово або NULL, якщо такого слова немає. Тому значенням рядка s1 є слово "Направо". Крім того функція запам'ятовує поточну позицію завершення пошуку, тобто вказівник на символ, з якого буде починатися пошук у наступній частині рядка. У нашому випадку цей вказівник адресуватиме решту рядка " Кроком руш!". Повторний виклик функції здійснюється так

   ```
   s1=strtok(NULL, " !");
   ```

   У решті рядка спочатку шукається символ, який не є розділяючим. Буде знайдена буква К, яка служить початком нового слова. Потім буде знайдений розділювач – пробіл і рядок набуде виду

   Кроком\0руш!

   Значенням рядка s1 буде слово "Кроком", а вказівник функції адресуватиме рядок "руш!". Знову можна викликати функцію

   ```
   s1=strtok(NULL, " !");
   ```

   Значенням s1 буде слово "руш". Для ще одного виклику

   ```
   s1=strtok(NULL, " !");
   ```

   отримаємо s1=NULL. Це є ознакою припинення викликів функції **strtok**.

   Організувавши за описаним алгоритмом пошук слів у рядку, ми заповнимо масив slova.

2. **Сортування масиву slova за алфавітом**. Для сортування скористаємося алгоритмом бульбашок. Оскільки сортування здійснюється на основі кодової таблиці символів, то

українські слова будуть розташовані «не зовсім» за алфавітом. Коди тільки українських букв (наприклад – є, ї) розташовані в окремій частині таблиці. Слова, які починаються з таких букв не будуть розташовуватися в алфавітному порядку.

3. **У масиві slova виконати пошук слів з підрядком «ах» та виконати перетворення над такими словами.** Створимо функцію

   int PerevirkaSlova(char *s1, char *s2).

   Через параметр s1 у функцію будемо передавати слово з масиву slova для перевірки. Функція буду повертати 1, якщо слово містить підрядок «ах», або 0 – в іншому випадку. Окрім того, через параметр s2 із функції у програму передаватиметься перетворене слово.

```
#include "stdafx.h"
#include "windows.h"
#include "string.h"
#define N 100                                   // Кількість слів у масиві slova
// Функція перевіряє, чи містить слово s1 підрядок «ах». Якщо так, то функція повертає 1,
// інакше – повертає 0. Крім того, через параметр s2 із функції у програму повертається
// перетворене слово, у якому перше входження «ах» замінене на «ого».
int PerevirkaSlova(char *s1, char *s2);     // Прототип функції
int _tmain(int argc, _TCHAR* argv[])
{       int i, j;
        char rydok[200];                        // Змінна для рядка слів
        char slova[N][11];                      // Оголошення масиву слів
        char s[8]=" .,:;!?";                    // Слово із розділяючих символів
        char s2[12];                            // Змінна для перетвореного слова,
                                                   у якому «ах» замінено на «ого»
        char *s1;                               // Допоміжна змінна – вказівник на слово
        int kilk=0;                     // Лічильник для підрахунку слів у початковому рядку
        int n=0;                                // Лічильник для кількості слів з «ах»
        printf ("Vvedit ryadok\n");
        gets(rydok);                            // Введення початкового рядка
        // Розбиття рядка на слова
        s1=strtok(rydok, s);
        while (s1!=NULL)
        {       strcpy(slova[kilk], s1);// Копіюємо знайдене слово s1 в чергове слово масиву
                s1=strtok(NULL, s);
                kilk++;
        }
        kilk--;                 // Остаточне значення кількості слів у початковому рядку
        // Виведення слів масиву slova
        printf ("Slova ryadka\n");
        for (i=0; i<=kilk; i++)
                printf("%s     ", slova[i]);
        printf ("\n");
        // Сортування слів за алфавітом методом бульбашок
        for (i=kilk-1; i>=0; i--)
                for (j=0; j<=i; j++)
                        if (strcmp(slova[j], slova[j+1])>0)    // Порівнюємо j-те слово з j+1-им
                        {       // Переставляємо слова
                                strcpy(s2, slova[j]);
                                strcpy(slova[j], slova[j+1]);
                                strcpy(slova[j+1], s2);
```

```
                        }
        // Виведення відсортованих слів
        printf ("Vidsortovani clova ryadka\n");
        for (i=0; i<=kilk; i++)
                printf("%s     ", slova[i]);
        printf ("\n");
        // Пошук та перетворення слів з підсловом "ах"
        printf ("Pary sliv: slovo z ax ta nove slovo\n");
        for (i=0; i<=kilk; i++)
                if (PerevirkaSlova(slova[i], s2))       // Виклик функції для пошуку слів з «ах»
                {       n++;
                        printf ("%15s    %15s\n", slova[i], s2);
                }
        printf ("Kilkist sliv z ax =%d\n", n);
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------------
int PerevirkaSlova(char *s1, char *s2)          // Функція для пошуку слів з «ах»
{       int i, k, m, flag=0;
        m=strlen(s1);                           // Обчислення довжини рядка s1
        for (i=0; i<m-1; i++)                   // Пошук підслова «ах» в s1
                if ((s1[i]==-96)&&(s1[i+1]==-27))           // Знайдено підслово «ах»
                {       flag=1;
                        break;                  // Припиняємо пошук
                }
        if (flag==0)                            // s1 не містить «ах»
                return 0;
        else                                    // s1 містить «ах»
                {       k=i;                    // Запам’ятовуємо місце, де зустілося «ах»
                        for (i=0; i<k; i++)     // Копіюємо в s2 всі символи рядка s1 до «ах»
                                s2[i]=s1[i];
                        // Заносимо в s2 слово «ого»
                        s2[k]=-82;
                        s2[k+1]=-93;
                        s2[k+2]=-82;
                        // Копіюємо в s2 решту символів s1, які зустрічаються після «ах»
                        for (i=k+2; i<=m; i++)
                                s2[i+1]=s1[i];
                        return 1;
                }
}
```

Прокоментуємо результати роботи програми. Спочатку був введений рядок із українських слів. Функція **strtok** дозволила виокремити із нього слова, які виведені на екран.

Стандартні функції мови С для роботи з рядками обробляють рядки, символи яких кодуються за допомогою одного байта. Це означає, що функції орієнтовані на кодову таблицю із 256 символів.

У програмі не включена локаль для виведення символів кирилиці, тобто немає виклику функції **setlocale** або іншого аналогу. Виклик функції **setlocale** призводить до не коректного відображення на екрані українських слів. Це пов’язане з тим, що середовище

Visual C++ 2010 використовує інше кодування символів кирилиці і воно має власні типи для символів.

```
Vvedit ryadok
У горах чути трах, бабах. Шарахнуло!!! Ах, як страшно. Невже? Ахахах...
Clova ryadka
У     горах     чути     трах     бабах     Шарахнуло     Ах     як     страшно
    Невже     Ахахах
Vidsortovani clova ryadka
Ах     Ахахах     Невже     У     Шарахнуло     бабах     горах     страшно
трах     чути     як
Pary sliv: slovo z ax ta nove slovo
         Ахахах              Ахогоах
      Шарахнуло          Шарогонуло
          бабах              бабого
          горах              горого
           трах               трого
Kilkist sliv z ax =5
Для продолжения нажмите любую клавишу . . .
```

Результати роботи програми.

На екран виведені відсортовані слова рядка. Спочатку в алфавітному порядку розташовані слова, що починаються із великих букв, а потім - з малих. Функція **stricmp** дозволяє не розрізняти символи верхнього і нижнього регістрів тільки для латинських букв, а не українських. Оскільки коди великих українських букв менші ніж у малих, то на екрані ми бачимо такий результат сортування. Для сортування українських слів треба використовувати не стандартний алгоритм.

Для пошуку підрядка в рядку використовується функція **strstr.** Однак вона не буде коректно працювати з українськими словами. У функції PerevirkaSlova ми реалізували наступний алгоритм. Слово «ах» складається із двох букв, коди яких -96, -27 відповідно. Тому в слові s1 ми шукали «ах» за кодами їхніх символів. Запам'ятовували місце розташування «ах» в слові s1. Створювали нове слово s2. У нього спочатку заносили символи s1, які були перед «ах». Потім в s2 заносили «ого» на основі кодів відповідних символів. Нарешті заносили в s2 частину рядка s1, яка розташована після «ах».

На екрані виведені пари слів: слово, що містить «ах», та перетворене слово, у якому «ах» замінене на «ого». Однак, у цей список не попало слово «Ах», бо в ньому «ах» не міститься. Коди символів А і а різні.

Задачі

1) Дана не порожня послідовність слів із латинських букв; сусідні слова відокремлені одне від одного комою, за останнім словом крапка. Визначити кількість слів, які:
   a) починаються з буква a;
   b) починаються і закінчуються одною і тою ж буквою;
   c) містять хоч би одну букву d.
2) Дано слово. Перевірити, чи всі букви в ньому різні.
3) Ціна товару записана в копійках. Наприклад, 317, 5005, 100 і т.д. Виразити ціну в гривнях і копійках. Наприклад, 3 гривні 17 копійок, 50 гривень 5 копійок, 1 гривня.
4) Надрукувати таблиці додавання та множення у шістнадцятковій системі числення.

☺ Індивідуальні завдання

Основний рівень

Дано рядок, довжина якого не перевищує **n** символів.

1. Усі повторні входження букви «а» у рядок замінити на букву «о». Якщо букви «а» у рядку немає, то вивести про це інформацію.

2. Підрахувати кількість входжень слова «мама» у рядок і вивести номери перших позицій цих входжень. Якщо цього слова у рядку немає, то вивести про це інформацію. Наприклад, рядок «мамамамама» містить 4 входження шуканого слова.
3. Підрахувати кількість входжень символів «+». «*» у рядок і вивести номери їх позицій. Якщо таких символів у рядку немає, то вивести про це інформацію.
4. Замінити усі символи «с» у рядку на «а». Якщо букв «с» у рядку немає, то вивести про це інформацію.
5. Замінити перший символ «к» рядка на символ «с», а останній символ «с» на «к». Якщо така заміна неможлива, то вивести про це інформацію.
6. Відшукати слово «ріпка» у рядку та замінити його на слово «колобок». Вивести номер початкової позиції виконаної заміни. Якщо така заміна неможлива, то вивести про це інформацію.
7. Підрахувати кількість відкритих та закритих круглих дужок у рядку та вивести номери їх позицій. Якщо дужок у рядку немає, то вивести про це інформацію.
8. Підрахувати кількість пробілів у рядку та вивести номери їх позицій. Перетворити рядок, викинувши пробіли. Якщо пробілів у початковому рядку немає, то вивести про це інформацію.
9. Підрахувати кількість відкритих та закритих квадратних дужок у рядку та вивести номери їх позицій. З'ясувати, чи є баланс дужок у рядку: для кожної дужки, яка відкривається, справа має бути дужка, що закривається. Якщо дужок у рядку немає, то вивести про це інформацію.
10. Замінити у рядку всі входження символа «н» на «нн». Якщо така заміна неможлива, то вивести про це інформацію.
11. Замінити у рядку всі входження «шч» на «щ». Якщо така заміна неможлива, то вивести про це інформацію.
12. Вивести номери позицій входження букви «и» у рядок. Вилучити з рядка всі входження цієї букви, крім останнього. Замінити останнє входження «и» на «е». Якщо букви «и» у рядку немає, то вивести про це інформацію.
13. Підрахувати кількість входжень символів «!». «?» у рядок і вивести номери їх позицій. Перетворити рядок, викинувши ці символи. Якщо таких символів у рядку немає, то вивести про це інформацію.
14. Підрахувати кількість відкритих та закритих фігурних дужок у рядку та вивести номери їх позицій. Вилучити усі символи, які розташовані всередині дужок. (Вважати, що для кожної відкритої дужки існує відповідна закрита, і між ними інших дужок немає). Якщо дужок у рядку немає, то вивести про це інформацію.
15. Підрахувати кількість малих латинських літер, що входять у рядок, та вивести номери їх позицій. Якщо таких літер у рядку немає, то вивести про це інформацію.

Підвищений рівень

1. Дано масив слів, і в кожному слові від 1 до 8 символів. Вивести ті слова, які є симетричними.
2. Дано масив слів, і в кожному слові від 1 до 8 малих латинських літер. Вивести ті слова, у яких букви впорядковані за алфавітом.
3. Дано масив слів, і в кожному слові від 1 до 8 символів. Вивести ті слова, у яких усі символи різні.
4. Дано масив слів, і в кожному слові від 2 до 10 букв. Вилучити в кожному слові всі повторні входження першої букви.
5. Дано масив слів, і в кожному слові від 3 до 10 малих латинських літер. У словах замінити всі входження «abc» на «def».
6. Дано масив слів, і в кожному слові від 2 до 10 малих латинських букв. У словах вилучити всі входження «th».

7. Дано масив слів і в кожному слові від 1 до 8 малих латинських літер. У всі слова після кожної букви «q» додати букву «u».
8. Дано масив слів, і в кожному слові від 1 до 8 малих латинських літер. У всі слова перед кожною буквою «b» вставити букву «a».
9. Дано масив слів, і в кожному слові від 2 до 10 малих латинських літер. У кожному слові поміняти місцями першу і останню літери.
10. Дано масив слів, і в кожному слові від 2 до 10 малих українських літер. Якщо слово парної довжини, то вилучити з нього дві середні літери, а непарної – середню літеру.
11. Дано масив слів, і в кожному слові від 1 до 8 малих латинських літер. Вивести ті слова, префікс яких є одним із слів «a», «ab», «abc».
12. Дано масив слів, і в кожному слові від 3 до 10 малих латинських літер. Вивести ті слова, суфікс яких співпадає зі словом «xyz».
13. Дано масив слів, і в кожному слові від 2 до 10 малих українських літер. У кожному слові вилучити всі повторні входження останньої літери, залишивши лише перше її входження.
14. Дано масив, слів і в кожному слові від 1 до 8 малих українських літер. У кожному слові кожну букву повторити (наприклад, зі слова «ох» отримати слово «ооххх»).
15. Дано масив слів, і в кожному слові від 2 до 10 малих латинських літер. Вивести ті слова, у яких голосні (a, e, i, o, u) чергуються з приголосними.

♖ Додаткові задачі

1. Дано рядок, який має наступний вигляд $d_1 \pm d_2 \pm d_3 \pm ... \pm d_n$ (d_i невід'ємні дійсні числа, $n>1$) і закінчується крапкою. Обчислити значення цієї суми.
2. Дано послідовність, що містить від 1 до 90 слів, у кожному з яких від 1 до 10 малих українських літер; між сусідніми словами не менше одного пробілу, після останнього слова крапка. Надрукувати ці слова за алфавітом.
3. Дано послідовність слів S_1, S_2, S_3, S_4, …, S_n. Перетворити цю послідовність так S_2, S_1, S_4, S_3,…
4. Дано слово з великих латинських літер. З'ясувати, чи є воно правильним записом римськими цифрами натурального числа від 1 до 999. Якщо є, то надрукувати це число арабськими цифрами (у десятковій системі).
5. Дано два символи: латинська літера (від a до h) і цифра (від 1 до 8), наприклад a2 або g5. Вважаючи їх координатами поля шахової дошки, на якому розташований ферзь, зобразити шахову дошку, заповнивши хрестиками всі поля, які «б'є» цей ферзь, а решту полів нулями.
6. Астрологи ділять рік на 12 періодів, і кожному з них відповідає один із знаків Зодіака:

20.1 – 18.2 - Водолій	23.7 – 22.8 - Лев
19.2 – 20.3 - Риби	23.8 – 22.9 - Діва
21.3 – 19.4 - Овен	23.9 – 22.10 - Терези
20.4 – 20.5 - Телець	23.10 – 22.11 - Скорпіон
21.5 – 21.6 - Близнюки	23.11 – 21.12 - Стрілець
22.6 – 22.7 - Рак	22.12 – 19.1 - Козеріг

   Написати програму, яка вводить дату деякого дня року і виводить назву відповідного знаку Зодіака.
8. Дано натуральне число n ($n \le 1000$). Записати це число словами (наприклад, сімнадцять, двісті п'ятдесят три, тисяча і т. п.).
9. Дано довільне натуральне число. Створити програму, яка виводить на екран це число римськими цифрами.
10. Дано римське число. Створити програму, яка виводить на екран це число у десятковій формі.

☝ Питання для самоконтролю

1. Дайте означення рядка.
2. Яка максимальна довжина рядка?
3. Вкажіть способи оголошення та ініціалізації рядкових змінних.
4. Вкажіть стандартні функції та процедури для роботи з рядками.
5. Як визначити довжину рядка?
6. Як отримати доступ до 5-го символу рядка?
7. Дано рядок s. Що міститься в s[0]?
8. У програмі потрібно ввести рядок з клавіатури. Яке з наступних оголошень змінної для рядка є правильним?
   1) char *st;
   2) char st[100];
   3) char st[];
   4) char *st[100];
9. Дано фрагмент програми:

   char *st = “На вулиці світить сонце.”

   Яке значення матиме кожен із наступних виразів?
   1) *st
   2) st[0]
   3) st[10]
   4) st[100]
   5) *st+10
   6) *(st+10)
   7) st[0]+10
   8) st[10]=”a”
   9) st[0]=’a’
   10) *(st+5)<’a’
   11) strlen(st)
   12) strlen(st+5)
10. Наведіть приклади оголошень та ініціалізації масивів рядків.
11. Наведіть приклади оголошень та ініціалізації масивів вказівників на рядки.

Практичне заняття № 9

*Тема: **Структури (записи)***

Студент повинен знати: означення структури та синтаксис її оголошення, способи ініціалізації структур та організації доступу до їх компонентів.

Теоретичні відомості

Структура – це структурований тип даних, що є об'єднанням фіксованого числа компонентів одного або декількох типів. Складові частини структури називаються полями або елементами.

Структура оголошується за синтаксисом:

struct < тег_структури >
{
< тип_поля1 > < ім'я_поля1 >;
< тип_поля2 > < ім'я_поля2>;
…….
< тип_поляk > < ім'я_поляk >;
};

Тут **struct** – службове слово; < тег_структури > - ім'я, яке позначає у програмі структуру даної форми; у фігурних дужках міститься список полів, кожне з яких є змінною.

Зауважимо, що структура – це тип. А змінна цього типу оголошується за синтаксисом

struct < тег_структури > < ім'я_структурної_змінної >

Для такої змінної виділяється пам'ять, розмір якої залежить від полів структури. Поля структури послідовно розташовуються в пам'яті.

Структурна змінна ініціалізується через надання значення її елементам. У мові С є дві операції для звертання до полів структурної змінної – крапка і «стрілка». Їхній синтаксис:

< ім'я_структурної_змінної >.< ім'я_поля >
< ім'я_вказівника_на_структуру > -> < ім'я_поля >

Приклад оголошення структури:

```
struct anketa
{       char prizv[20];
        int vik;
        double zarplata;
};
```

Оголошення структурної змінної і вказівника на структуру:

```
struct anketa kozak;
struct anketa *ps;
```

Ініціалізація структури при оголошенні:

```
struct anketa kozak = {"Kovalenko", 45, 5670.50};
```

Звертання до полів структури:

kozak.prizv kozak.vik ps->prizv ps->vik

Ці звертання є змінними, яким можна надавати значення та обробляти їх.

Приклад 1

Створити масив структур із полями: прізвище учня і три оцінки. Вивести прізвища учнів, які мають хоча б одну двійку.

Аналіз задачі

Задача є простою. Доцільно створити функції для занесення та виведення даних структурних змінних, пошуку в масиві даних про учнів, що мають двійку. Пошук є послідовним і елементарно реалізується для масиву. Алгоритм очевидний.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 10                    // Кількість елементів у масиві
// Перевизначення типу структури. Новий тип називається uchen
typedef struct zapys
        {       char prizv[20];
                int ocinka[3];
        } uchen;
// Функція для заповнення полів структури. Параметр p є вказівником на структуру. Через
// нього у функцію передається незаповнена структура, а повертається – заповнена.
void StvorenZapys(uchen *p);
// Функція для виведення полів структури. Параметр p є вказівником на структуру. Через
// нього у функцію передається структура, поля якої виводяться на екран
void DrucZapys(uchen *p);
// Функція здійснює пошук учнів у масиві, які мають двійки. Через параметр mas у
// функцію передається вказівник на масив, а через параметр n – розмірність масиву.
void PoiskZapys(uchen mas[], int n);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i;
        uchen massiv[N];                // Оголошення масиву структур
        // Заповнення масиву структур
        for (i=0; i<N; i++)
                StvorenZapys(&massiv[i]);
        // Виведення масиву
        printf ("Виведення інформації про учнів\n");
        for (i=0; i<N; i++)
                DrucZapys(&massiv[i]);
        // Пошук учнів з двійками
        printf ("Прізвища учнів, що мають двійки\n");
        PoiskZapys(massiv, N);
        system("pause");
        return 0;
}
//-----------------------------------------------------------------------------------------------------------------
void StvorenZapys(uchen *p)
{       printf ("Введіть прізвище учня   ");
        scanf ("%s", p->prizv);
        printf ("Введіть першу оцінку учня   ");
        scanf ("%d", &(p->ocinka[0]));
        printf ("Введіть другу оцінку учня   ");
        scanf ("%d", &(p->ocinka[1]));
        printf ("Введіть третю оцінку учня   ");
        scanf ("%d", &(p->ocinka[2]));
}
```

```
//-----------------------------------------------------------------------------------------------------------------
void DrucZapys(uchen *p)
{
	printf ("%20s%5d%5d%5d\n", p->prizv, p->ocinka[0], p->ocinka[1], p->ocinka[2]);
}
//-----------------------------------------------------------------------------------------------------------------
void PoiskZapys(uchen mas[], int n)
{	int i, d=0;		// Змінна d для підрахунку учнів, які не мають двійок
	for (i=0; i<n; i++)
		if ((mas[i].ocinka[0]==2)||(mas[i].ocinka[1]==2)||(mas[i].ocinka[2]==2))
			printf ("%20s", mas[i].prizv);		// Учень, який має дійку
		else	d++;
	if (d==n)
		printf ("Учнів з двійками немае\n");
}
```

```
Введіть прізвище учня   Kovalenko
Введіть першу оцінку учня   4
Введіть другу оцінку учня   4
Введіть третю оцінку учня   5
Введіть прізвище учня   Petrenko
Введіть першу оцінку учня   3
Введіть другу оцінку учня   3
Введіть третю оцінку учня   3
Введіть прізвище учня   Sokol
Введіть першу оцінку учня   3
Введіть другу оцінку учня   2
Введіть третю оцінку учня   3
Введіть прізвище учня   Somyk
Введіть першу оцінку учня   5
Введіть другу оцінку учня   5
Введіть третю оцінку учня   5
Введіть прізвище учня   Romanov
Введіть першу оцінку учня   2
Введіть другу оцінку учня   2
Введіть третю оцінку учня   2
Введіть прізвище учня   Rak
Введіть першу оцінку учня   5
Введіть другу оцінку учня   5
Введіть третю оцінку учня   5
Введіть прізвище учня   Pavuk
Введіть першу оцінку учня   2
Введіть другу оцінку учня   3
Введіть третю оцінку учня   2
Введіть прізвище учня   Mamchur
Введіть першу оцінку учня   3
Введіть другу оцінку учня   4
Введіть третю оцінку учня   3
Введіть прізвище учня   Sova
Введіть першу оцінку учня   5
Введіть другу оцінку учня   4
Введіть третю оцінку учня   5
Введіть прізвище учня   Strixa
Введіть першу оцінку учня   2
Введіть другу оцінку учня   2
Введіть третю оцінку учня   3
Виведення інформації про учнів
          Kovalenko    4    4    5
           Petrenko    3    3    3
              Sokol    3    2    3
              Somyk    5    5    5
            Romanov    2    2    2
                Rak    5    5    5
              Pavuk    2    3    2
            Mamchur    3    4    3
               Sova    5    4    5
             Strixa    2    2    3
Прізвища учнів, що мають двійки
              Sokol             Romanov               Pavuk               Strixa
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 2

Створити масив структур із полями: прізвище, адреса і заборгованість за квартплату. Вивести в алфавітному порядку інформацію про тих абонентів, які мають найбільшу заборгованість. Вивести інформацію за спаданням заборгованості про всіх абонентів, що мають заборгованість і проживають на заданій вулиці.

Аналіз задачі та алгоритм

Розв'язання задачі складається з таких кроків:

1. Створити масив структур.
2. Знайти у масиві величину найбільшої заборгованості.
3. Знайти у масиві інформацію про всіх абонентів, що мають найбільшу заборгованість. Доцільно цю інформацію запам'ятати у допоміжному масиві.
4. Упорядкувати допоміжний масив в алфавітному порядку (за прізвищами).
5. Знайти в початковому масиві інформацію про всіх абонентів, що мають заборгованість і проживають на заданій вулиці. Доцільно цю інформацію запам'ятати у допоміжному масиві.
6. Упорядкувати допоміжний масив за спаданням заборгованості.

Оскільки адреса складається з назви вулиці, номера будинку і номера квартири, то нам потрібно використати вкладення структур.

У пунктах 3 і 5 можна не використовувати допоміжний масив. Можна упорядковувати весь масив за певною ознакою, а потім шукати в ньому потрібну інформацію. Однак сортування всього масиву займає більше часу ніж якоїсь його частини. З іншого боку, допоміжний масив вимагає додаткової пам'яті. Якщо початковий масив дуже великий, то доцільно використати допоміжний масив або іншу організацію даних.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
#define N 15
struct adr              // Оголошення структури для адреси
        {       char vulysja[20];           // Назва вулиці
                int nomerbud;               // Номер будинку
                int nomerkvart;             // Номер квартири
        };
// Перейменування типу структури для абонента
typedef struct zapys
        {       char prizv[20];             // Прізвище
                struct adr adresa;          // Адреса. Вкадення структури
                double borg;                // Заборгованість
        } abonent;                  // Новий тип структури
```

// Функція для заповнення полів структури. Параметр p є вказівником на структуру. Через нього у функцію передається незаповнена структура, а повертається – заповнена.

void StvorenZapys(abonent *p);

// Функція для виведення полів структури. Параметр p є вказівником на структуру. Через нього у функцію передається структура, поля якої виводяться на екран

void DrucZapys(abonent *p);

// Функція для сортування масиву за алфавітом прізвищ. Параметр mas є вказівником на масив структур, а n – кількість елементів у масиві.

void SortAlfavit(abonent mas[], int n);

// Функція для сортування масиву за спаданням боргу. Параметр mas є вказівником на масив структур, а n – кількість елементів у масиві.

void SortBorg(abonent mas[], int n);

```
int _tmain(int argc, _TCHAR* argv[])
{	setlocale (LC_ALL,"RUS");
	int i;
	abonent massiv[N];		// Оголошення масиву структур для абонентів
	abonent dmassiv[N];		// Оголошення допоміжного масиву структур
	// Заповнення масиву структур
	for (i=0; i<N; i++)
		StvorenZapys(&massiv[i]);
	// Виведення масиву
	printf ("Виведення інформації про абонентів\n");
	for (i=0; i<N; i++)
		DrucZapys(&massiv[i]);
	// Пошук максимальної заборгованості
	double maxborg=0;		// Початкове значення найбільшої заборгованості
	for (i=0; i<N; i++)		// Пошук найбільшої заборованості
		if (massiv[i].borg > maxborg)
			maxborg=massiv[i].borg;
	printf ("maxborg=%lf\n", maxborg);
	// Пошук абонентів з максимальною заборгованістю і
	// заповнення допоміжного масиву
	int d=0;	// Змінна для визначення кількості елементів у допоміжному масиві
	for (i=0; i<N; i++)
		if (massiv[i].borg == maxborg)
		{	d++;
			// Копіювання елемента massiv[i] в dmassiv[d-1]
			strcpy(dmassiv[d-1].prizv, massiv[i].prizv);
			strcpy(dmassiv[d-1].adresa.vulysja, massiv[i].adresa.vulysja);
			dmassiv[d-1].adresa.nomerbud = massiv[i].adresa.nomerbud;
			dmassiv[d-1].adresa.nomerkvart = massiv[i].adresa.nomerkvart;
			dmassiv[d-1].borg = massiv[i].borg;
		}
	// Виведення допоміжного масиву
	printf ("Виведення інформації про абонентів, що мають найбільшу
                                                                    заборгованість\n");
	for (i=0; i<d; i++)
		DrucZapys(&dmassiv[i]);
	// Сортування допоміжного масиву за алфавітом
	if (d>1)
		SortAlfavit(dmassiv, d);
	// Виведення відсортованого масиву
	printf ("Виведення за алфавітом абонентів, що мають найбільшу заборгованість\n");
	for (i=0; i<d; i++)
		DrucZapys(&dmassiv[i]);
	char vul[20];			// Змінна для назви вулиці
	printf ("Введіть назву вулиці\n");
	scanf("%s", vul);
	// Пошук абонентів, що проживають на даній вулиці та мають заборгованість.
	// Занесення їх в допоміжний масив
	d=0;	// Змінна для визначення кількості елементів у допоміжному масиві
	for (i=0; i<N; i++)
		if ((strcmp(massiv[i].adresa.vulysja, vul) == 0) && (massiv[i].borg > 0))
		{	d++;
```

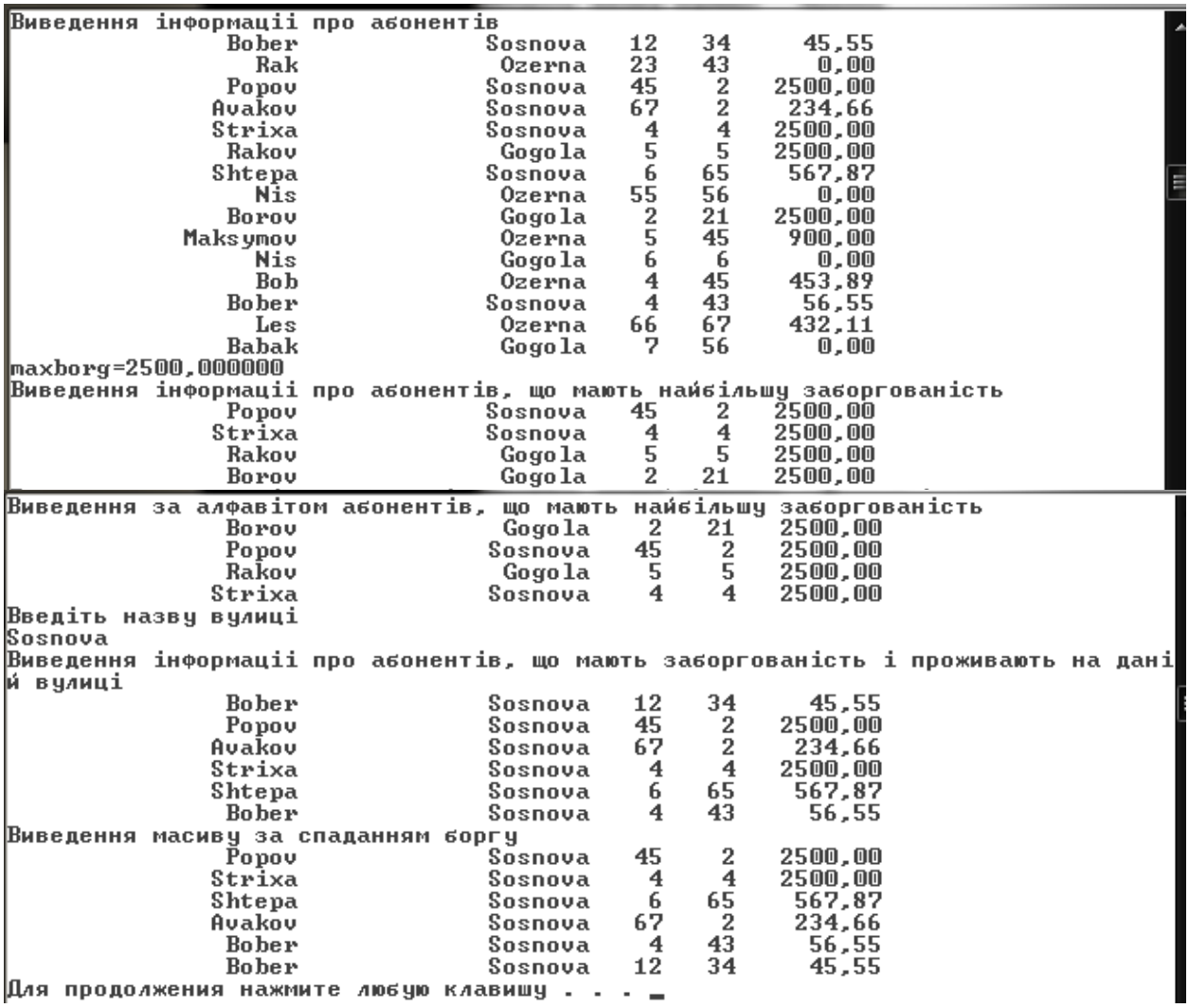
```
                        // Копіювання елемента massiv[i] в dmassiv[d-1]
                        strcpy(dmassiv[d-1].prizv, massiv[i].prizv);
                        strcpy(dmassiv[d-1].adresa.vulysja, massiv[i].adresa.vulysja);
                        dmassiv[d-1].adresa.nomerbud = massiv[i].adresa.nomerbud;
                        dmassiv[d-1].adresa.nomerkvart = massiv[i].adresa.nomerkvart;
                        dmassiv[d-1].borg = massiv[i].borg;
                }
        // Виведення допоміжного масиву
        printf ("Виведення інформації про абонентів, що мають заборгованість і
                                                                проживають на даній вулиці\n");
        for (i=0; i<d; i++)
                DrucZapys(&dmassiv[i]);
        // Сортування допоміжного масиву за спаданням заборгованості
        if (d>1)
                SortBorg(dmassiv, d);
        // Виведення відсортованого масиву
        printf ("Виведення масиву за спаданням боргу\n");
        for (i=0; i<d; i++)
                DrucZapys(&dmassiv[i]);
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------
void StvorenZapys(abonent *p)
{       // Заповнення інформації про абонента. Заповнена структура передається у
        // програму через вказівник p.
        printf ("Введіть прізвище   ");
        scanf ("%s", p->prizv);
        printf ("Введіть назву вулиці   ");
        scanf ("%s", p->adresa.vulysja);
        printf ("Введіть номер будинку   ");
        scanf ("%d", &(p->adresa.nomerbud));
        printf ("Введіть номер квартири   ");
        scanf ("%d", &(p->adresa.nomerkvart));
        printf ("Введіть заборгованість   ");
        scanf ("%lf", &(p->borg));
}
//-------------------------------------------------------------------------------------------------------------
void DrucZapys(abonent *p)
{       // Виведення структури на екран
        printf ("%20s%20s%5d%5d%10.2lf\n", p->prizv, p->adresa.vulysja,
                                        p->adresa.nomerbud, p->adresa.nomerkvart, p->borg);
}
//-------------------------------------------------------------------------------------------------------------
void SortAlfavit(abonent mas[], int n)
{       // Сортування масиву методом бульбашок за алфавітом прізвищ
        int i, j;
        abonent z;                      // Структурна змінна
        for (i=n-2; i>=0; i--)          // Сортування методом бульбашок
                for (j=0; j<=i; j++)
                        if (strcmp(mas[j].prizv, mas[j+1].prizv) > 0)
                        {       // Перестановка елементів mas[j] та mas[j+1]
```

```
                        // Елемент mas[j] запам’ятовуємо в змінній z
                        strcpy(z.prizv, mas[j].prizv);
                        strcpy(z.adresa.vulysja, mas[j].adresa.vulysja);
                        z.adresa.nomerbud = mas[j].adresa.nomerbud;
                        z.adresa.nomerkvart = mas[j].adresa.nomerkvart;
                        z.borg = mas[j].borg;
                        // Елемент mas[j+1] копіюємо в mas[j]
                        strcpy(mas[j].prizv, mas[j+1].prizv);
                        strcpy(mas[j].adresa.vulysja, mas[j+1].adresa.vulysja);
                        mas[j].adresa.nomerbud = mas[j+1].adresa.nomerbud;
                        mas[j].adresa.nomerkvart = mas[j+1].adresa.nomerkvart;
                        mas[j].borg = mas[j+1].borg;
                        // Копіюємо z в mas[j]
                        strcpy(mas[j+1].prizv, z.prizv);
                        strcpy(mas[j+1].adresa.vulysja, z.adresa.vulysja);
                        mas[j+1].adresa.nomerbud = z.adresa.nomerbud;
                        mas[j+1].adresa.nomerkvart = z.adresa.nomerkvart;
                        mas[j+1].borg = z.borg;
                }
}
//-------------------------------------------------------------------------------------------------------------------
void SortBorg(abonent mas[], int n)
{       // Сортування масиву методом бульбашок за спаданням заборгованості
        int i, j;
        abonent z;                      // Структурна змінна
        for (i=n-2; i>=0; i--)
                for (j=0; j<=i; j++)
                        if (mas[j].borg < mas[j+1].borg)
                        {       // Перестановка елементів mas[j] та mas[j+1]
                                // Елемент mas[j] запам’ятовуємо в змінній z
                                strcpy(z.prizv, mas[j].prizv);
                                strcpy(z.adresa.vulysja, mas[j].adresa.vulysja);
                                z.adresa.nomerbud = mas[j].adresa.nomerbud;
                                z.adresa.nomerkvart = mas[j].adresa.nomerkvart;
                                z.borg = mas[j].borg;
                                // Копіюємо mas[j+1] в mas[j]
                                strcpy(mas[j].prizv, mas[j+1].prizv);
                                strcpy(mas[j].adresa.vulysja, mas[j+1].adresa.vulysja);
                                mas[j].adresa.nomerbud = mas[j+1].adresa.nomerbud;
                                mas[j].adresa.nomerkvart = mas[j+1].adresa.nomerkvart;
                                mas[j].borg = mas[j+1].borg;
                                // Копіюємо z в mas[j]
                                strcpy(mas[j+1].prizv, z.prizv);
                                strcpy(mas[j+1].adresa.vulysja, z.adresa.vulysja);
                                mas[j+1].adresa.nomerbud = z.adresa.nomerbud;
                                mas[j+1].adresa.nomerkvart = z.adresa.nomerkvart;
                                mas[j+1].borg = z.borg;
                        }
}
```

```
Введіть прізвище   Bober
Введіть назву вулиці   Sosnova
Введіть номер будинку   12
Введіть номер квартири   34
Введіть заборгованість   45,55
Введіть прізвище   Rak
Введіть назву вулиці   Ozerna
Введіть номер будинку   23
Введіть номер квартири   43
Введіть заборгованість   0
Введіть прізвище   Popov
Введіть назву вулиці   Sosnova
Введіть номер будинку   45
Введіть номер квартири   2
Введіть заборгованість   2500
Введіть прізвище   Avakov
Введіть назву вулиці   Sosnova
Введіть номер будинку   67
Введіть номер квартири   2
Введіть заборгованість   234,66
Введіть прізвище   Strixa
Введіть назву вулиці   Sosnova
Введіть номер будинку   4
Введіть номер квартири   4
Введіть заборгованість   2500
```

```
Введіть прізвище   Rakov
Введіть назву вулиці   Gogola
Введіть номер будинку   5
Введіть номер квартири   5
Введіть заборгованість   2500
Введіть прізвище   Shtepa
Введіть назву вулиці   Sosnova
Введіть номер будинку   6
Введіть номер квартири   65
Введіть заборгованість   567,87
Введіть прізвище   Nis
Введіть назву вулиці   Ozerna
Введіть номер будинку   55
Введіть номер квартири   56
Введіть заборгованість   0
Введіть прізвище   Borov
Введіть назву вулиці   Gogola
Введіть номер будинку   2
Введіть номер квартири   21
Введіть заборгованість   2500
Введіть прізвище   Maksymov
Введіть назву вулиці   Ozerna
Введіть номер будинку   5
Введіть номер квартири   45
Введіть заборгованість   900
```

```
Введіть прізвище   Nis
Введіть назву вулиці   Gogola
Введіть номер будинку   6
Введіть номер квартири   6
Введіть заборгованість   0
Введіть прізвище   Bob
Введіть назву вулиці   Ozerna
Введіть номер будинку   4
Введіть номер квартири   45
Введіть заборгованість   453,89
Введіть прізвище   Bober
Введіть назву вулиці   Sosnova
Введіть номер будинку   4
Введіть номер квартири   43
Введіть заборгованість   56,55
Введіть прізвище   Les
Введіть назву вулиці   Ozerna
Введіть номер будинку   66
Введіть номер квартири   67
Введіть заборгованість   432,11
Введіть прізвище   Babak
Введіть назву вулиці   Gogola
Введіть номер будинку   7
Введіть номер квартири   56
Введіть заборгованість   0
```



Результат роботи програми.

Задачі

1. Задано два масиви структур. Структури першого масиву містять відомості про прізвища людей та їхні адреси, другого – про прізвища та номери телефонів. Скласти програму, що за цими двома масивами формує третій масив, структури якого мають такі поля: прізвище, адреса, номер телефону.
2. Створити два масиви, що містять відомості про прізвища та адреси людей. Структури, які є у першому масиві та яких немає у другому масиві, скопіювати до третього масиву.

☺ Індивідуальні завдання

Основний рівень

1. Створити масив структур із полями: прізвище та рік народження. Надрукувати прізвища тих, хто народився в 1980 році.
2. Створити масив, який містить дані про учнів: прізвище, клас, номер школи і середній бал оцінок за чверть. З'ясувати, скільки учнів мають середній бал не нижчий за 8.
3. Створити масив, який містить дані про книги. Відомості про кожну книгу: прізвище автора, назва книги та рік видання. Вивести на екран усі книги вказаного автора та їхні роки видання.

4. Дані про автомобіль складаються з його марки, номера та прізвища господаря. Створити масив, який містить дані про декілька автомобілів. Вивести інформацію про господарів автомобілів певної марки.
5. Створити масив структур із полями: прізвище учня, число та місяць народження. Надрукувати список учнів, які народилися влітку.
6. Створити масив, який містить різні дати. Кожна з цих дат складається з числа, місяця та року. Знайти дати з найменшим номером року.
7. Створити масив, який містить дані про книги. Відомості про кожну книгу: прізвище автора, назва книги та рік видання. Визначити, чи є книга з назвою ”Інформатика”, і якщо є, то повідомити прізвище автора та рік видання.
8. Створити масив, який містить дані про куби: розмір кожного куба (довжина ребра), його колір та матеріал. Вивести інформацію про дерев’яні куби.
9. Створити масив структур, який містить дані про учнів школи. Поля структури: прізвище, ім’я, клас та номер школи. Надрукувати дані про учнів, прізвища яких найкоротші.
10. Створити масив структур із полями: прізвище учня, число та місяць народження. Надрукувати список учнів, які народилися взимку.
11. Створити масив, який містить дані про учнів: прізвище, клас, номер школи і гурток, у роботі якого учень бере участь. З’ясувати, скільки учнів і які саме відвідують вказаний гурток.
12. Створити масив структур, який містить дані про системні блоки: розмір, колір та ціна. Вивести на екран дані про системні блоки певної ціни.
13. Створити масив структур із полями: назва книги, автор та рік видання. Надрукувати дані про книги, які видані з n-го року.
14. На Олімпійських Іграх брали участь гімнасти різних країн. Відомості про кожного складаються з імені, прізвища, країни та кількості балів. Створити масив, який містить відомості про спортсменів. Вивести на екран інформацію про представників України.
15. У змаганнях зі спортивної гімнастики беруть участь декілька спортсменів. Вони виконують вправи на 4 снарядах. Масив містить відомості про кожного спортсмена: прізвище та оцінки на кожному снаряді. Вивести прізвища тих спортсменів, які в сумі отримали найбільше балів.

Підвищений рівень

1. Масив містить відомості про автомобілі: марка, країна, вартість. Вивести за зростанням вартості інформацію про авто, які випускаються в країні К.
2. Відомості про учнів складаються з імені, прізвища та дати народження (дата містить число, місяць та рік). Вивести на екран відомості за алфавітом прізвищ про тих учнів, які народилися в заданому місяці.
3. Створити масив структур із полями: місяць, число. Вивести на екран за зростанням спочатку всі літні дати, потім усі зимові, а інші замінити на нулі і не виводити.
4. Створити масив структур із полями: прізвище, ім’я, вік. Написати програму, яка виводить на екран за алфавітом прізвища людей, вік яких найбільший.
5. Створити масив структур із полями: прізвище учасника змагань та його результат (час у секундах). Надрукувати прізвища та результати переможців (1, 2 та 3 місця).
6. Створити масив структур із полями: прізвище студента та три оцінки за сесію. Вивести назву предмета, який здано найкраще. Потім вивести за алфавітом прізвищ інформацію про те, яку оцінку отримав кожний студент із цього предмету.
7. Створити масив структур із полями: прізвище студента та три оцінки за сесію. Знайти двієчників (тих, хто має хоча б 1 двійку) та вивести про них інформацію на екран за алфавітом прізвищ.
8. Створити масив структур із полями: ім’я учня, вік, зріст. Знайти найвищого учня, учня середнього зросту та відсортувати масив за спаданням зросту.

9. Створити масив структур із полями: прізвище, рік народження та номер школи. Вивести за алфавітом прізвищ інформацію про учнів, які навчаються у школі з даним номером, та, яким у цьому році виповнилося 16 років.
10. Створити масив структур із полями: прізвище учня та три оцінки. Визначити якісний показник успішності (кількість 4 та 5 в %) та упорядкувати масив за спаданням середнього балу учнів.
11. Дані про учня складаються з його імені, прізвища та класу, у якому він навчається. Створити масив, який містить відомості про учнів школи. З'ясувати, чи є у школі учні з однаковим прізвищем. Упорядкувати масив за зростанням номеру класу, а у межах одного класу упорядкувати за алфавітом прізвищ.
12. Створити масив структур із полями: прізвище, телефон, адреса. Вивести на екран за алфавітом прізвищ інформацію про тих абонентів, телефон яких починається з цифри 2.
13. Створити масив, який містить дані про книги. Відомості про кожну книгу: прізвище автора, назва книги та рік видання. Вивести на екран за зростанням року видання інформацію про книги, назва яких починається з "Інф". Якщо таких книг немає, то повідомити про це.
14. У змаганнях зі спортивної гімнастики беруть участь декілька спортсменів. Вони виконують вправи на 4 снарядах. Масив містить відомості про кожного спортсмена: прізвище та оцінки на кожному снаряді. Вивести на екран інформацію про спортсменів за спаданням сум отриманих балів.
15. Створити масив структур із полями: ім'я, вік, стать. Написати програму, яка виводить на екран інформацію про чоловіків за зменшенням віку.

♖ Додаткові задачі

1. Створити масив структур для збереження бібліотечного каталогу, у якому є дані про книги, журнали та газети. Про книгу відомі її назва, прізвище автора та рік видання, про журнал – його назва, номер, рік видання та перелік статей із прізвищами авторів. Газети ідентифікуються так: назва газети, її номер, дата виходу, перелік статей із прізвищами авторів. У створеному масиві здійснити пошук робіт автора, прізвище якого введено із клавіатури. Із масиву вилучити всі газети, видані до 1995 року.
2. Створити базу даних про студентів деякого вузу. Інформація про студента: прізвище, ім'я, по-батькові, стать, вік, курс. Розробити програму, яка створює базу і друкує наступні дані:
    a) номер курсу, на якому найбільший процент чоловіків;
    b) найпоширеніше чоловіче та жіноче ім'я;
    c) прізвища (в алфавітному порядку) та ініціали всіх студенток, вік і по-батькові яких є одночасно найпоширенішими.

☝ Питання для самоконтролю

1. Що таке структура?
2. Для представлення якої інформації можна використовувати структури?
3. Наведіть синтаксис оголошення типу структури.
4. Як звернутися до поля структури?
5. Яким може бути тип поля структури?
6. Які є способи ініціалізації структури?

Практичне заняття № 10

*Тема: **Текстові файли***

Студент повинен знати: поняття фізичного та логічного файлів, типи файлів і оголошення файлових змінних, встановлення відповідності між фізичним і логічним файлами, стандартні функції для роботи з файлами, зчитування і запис текстових файлів.

Теоретичні відомості

Файл – це іменована область на зовнішньому носії інформації, що містить довільні дані. Файл у такому розумінні називають **фізичним** файлом, тобто таким, що існує фізично на матеріальному носії інформації. Такий файл є послідовністю байтів, ідентифікується іменем та має певний розмір. З іншого боку, файл – це структура даних, що використовується у програмуванні. У такому розумінні файл називають **логічним,** тобто таким, що існує у певній програмі як абстракція. Логічний файл є послідовністю значень певного типу, тобто він складається з однотипних компонентів і є структурованим елементом даних. Оскільки компоненти файлу належать до одного типу, то структура логічного файлу нагадує структуру масиву. Але існують суттєві розбіжності між цими структурами даних:

1. Під час оголошення масиву треба вказати кількість його елементів. Під час оголошення файлової змінної розмір файлу невідомий;
2. Розмір масиву, на відміну від розміру файлу, не може змінюватися під час роботи з ним;
3. Для доступу до елементів масиву застосовують індексацію, а для доступу до компонентів файлу – вказівник на поточний компонент;

Файли класифікують за типом компонентів і за методом доступу до них. За типом компонентів розрізняють **текстові** та **бінарні** (двійкові) файли, а за методом доступу – файли **послідовного** і **прямого** доступу. Текстові файли призначені для збереження текстів, а бінарні файли використовують для збереження даних різних типів.

Компілятор мови С немає вбудованих засобів для введення/виведення даних. Тому обмін даних як зі зовнішніми (консольними) пристроями, так і з дисковими файлами реалізовано через відповідні набори бібліотечних функцій.

Бібліотеки більшості систем програмування мови С підтримують функції, що дають змогу здійснювати операції введення/виведення даних на трьох рівнях:

- високорівневе, так зване потокоорієнтоване введення/виведення;
- введення/виведення низького рівня;
- обмін даними з консольними пристроями.

Високорівневе введення/виведення є потокоорієнтованим, тому що всі файли та дані з пристроїв розглядаються як неструктуровані набори байтів – потоки. Прототипи функцій потокорієнтованого буферизованого обміну даними містяться в файлі stdio.h. Ці функції підтримуються стандартом мови С. Функції низькорівневого введення/виведення базуються на засобах обміну, що властиві кожній конкретній операційній системі, тому вони не належать до стандартизованих. Прототипи функцій консольного введення/виведення оголошені в файлі conio.h. Ми розглядаємо лише високорівневе введення/виведення даних.

У мові С немає чіткого поділу файлів на види. Однак існує можливість працювати з файлами послідовного та прямого доступу. Файли послідовного доступу є текстовими, а прямого – бінарними.

Бібліотечні функції для роботи з файлами на рівні потоків дозволяють обробляти дані різних розмірів і форматів, забезпечуючи при цьому буферизоване введення/ виведення. У мові С потік – це файл або зовнішній пристрій разом з наданими засобами буферизації.

При роботі з потоками можна виконувати наступні дії:

- відкривати і закривати потоки (зв'язувати вказівники на потоки з конкретними файлами);
- вводити і виводити: символ, рядок, форматні дані, порцію даних довільної довжини;
- аналізувати помилки потокового введення/виведення і умову досягнення кінця потоку;
- керувати буферизацією потоку і розміром буфера;
- отримувати і встановлювати вказівник (індикатор) поточної позиції в потоці.

Робота з файлами (потоками) у мові С складається з таких етапів:

1. оголошення вказівника на потік;
2. відкриття файлу;
3. обробка файлу;
4. закриття файлу.

Із кожним відкритим файлом система програмування зв'язує спеціальний **вказівник** (вказівник поточної позиції файлу), який вказує на той (поточний) компонент файлу, над яким буде виконано операцію читання чи запису. Як тільки операція виконана, вказівник автоматично зсувається на наступний компонент, тобто можна виконувати операцію вже з наступним компонентом. При відкритті файлу вказівник встановлюється на початок файлу, тобто на компонент із порядковим номером 0.

Оголошення вказівника на потік здійснюється за синтаксисом:

FILE *<ім'я_вказівника>;

Наприклад,

FILE *fp;

Визначення структурного типу FILE міститься в файлі stdio.h. В структурі FILE містяться компоненти, з допомогою яких ведеться робота з потоком, зокрема вказівник на буфер, вказівник (індикатор) поточної позиції в потоці та інша інформація.

При відкритті потоку в програму повертається вказівник на потік, що є вказівником на тип FILE. Цей вказівник ідентифікує потік у всіх наступних операціях.

Створення потоку реалізує функція відкриття файлу, прототип якої має вигляд:

FILE * fopen(char * filename, char * mode);

Функція створює новий потік і пов'язує його з фізичним файлом, заданим іменем filename. Параметр mode задає режим відкриття потоку. За умови успішного відкриття потоку функція повертає адресу (вказівник) на структуру FILE. Якщо ж потік відкрити не вдалося, то структура FILE не створюється, а fopen() повертає вказівник NULL.

Залежно від параметра mode потік можна відкрити в текстовому або двійковому (бінарному) режимі. Значення цього параметра подані в таблиці 10.1.

Таблиця 10.1. Режими відкриття файлу (потоку).

Режим для текстових файлів	**Режим для двійкових файлів**	**Призначення режиму**
"r"	**"rb"**	Існуючий файл відкривається тільки для читання. Якщо файлу не існує, то фіксується помилка і fopen повертає значення NULL.
"w"	**"wb"**	Відкривається новий файл тільки для запису. Якщо файл уже існував, то його вміст витирається, файл створюється заново.
"a"	**"ab"**	Відкривається файл (або створюється, якщо файлу немає) для добавлення в його кінець нової інформації.
"r+"	**"rb+","r+b"**	Існуючий файл відкривається як для читання, так і для запису в будь-якому місці файлу. Однак в цьому режимі неможливий запис в кінець файлу, тобто не можна збільшити файлу. Якщо файлу не існує, то фіксується помилка і fopen повертає значення NULL.

“w+”	**“wb+”,“w+b”**	Відкривається новий файл як для запису, так і для наступних багатократних виправлень. Якщо файл уже існував, то його вміст витирається, файл створюється заново. Наступні після відкриття файлу запис і читання з нього допустимі в будь-якому місці файлу, зокрема дозволений запис в кінець файлу.
“a+”	**“ab+”,“a+b”**	Відкривається файл (або створюється, якщо файлу немає) і стає доступним для змін, тобто для запису і для читання в будь-якому місці. На відміну від попереднього режиму, можна добавляти в кінець файлу нову інформацію без втрати початкового файлу.

Закриття потоку здійснює функція fclose, прототип якої має вигляд:

int fclose(FILE * fp);

Функція має один параметр fp – вказівник на потік, який треба закрити. При успішному закритті потоку (файлу) функція fclose() повертає значення 0, а при невдачі – повертає макроконстанту EOF, оголошену в stdio.h. Створений програмно новий файл треба обов’язково закрити, бо при закритті формується ознака кінця файлу. Далі можна знову відкривати файл та обробляти дані.

На початку виконання кожної С-програми автоматично відкриваються три стандартні потоки:

- stdin – стандартний потік введення, який за замовчуванням пов’язується з клавіатурою;
- stdout – стандартний потік виведення, який найчастіше пов’язується з виведенням даних на екран;
- stderr – стандартний потік повідомлень про помилки, який теж здебільшого скеровується на екран.

Стандартні потоки є текстовими файлами. Для роботи з ними використовуються функції:

- getchar() / putchar() – введення/виведення окремих символів;
- gets() / puts() – введення/виведення рядків;
- scanf() / printf() – введення/виведення в режимі форматування даних.

Аналоги цих функцій є для довільних текстових файлів. Для перенаправлення потоків можна використати функцію freopen().

Для зчитування одного символу з файлу використовуються функції:

int fgetc(FILE * fp);

int getc(FILE * fp);

Функції повертають код зчитаного символу. Якщо ж читання недоступне, то повертається макроконстанта EOF.

Запис заданого символу sym у потік виведення fp виконують функції:

int fputc(int sym, FILE * fp);

int putc(int sym, FILE * fp);

Функції повертають код записаного символу. Якщо ж запис неможливий, то повертається макроконстанта EOF.

Зчитування рядків з потоку введення виконує функція

char * fgets(char * str, int max, FILE * fp);

Параметр str вказує на масив символів, у який буде записаний рядок із потоку fp. Параметр max задає максимальну кількість символів (з нуль-символом включно), які будуть зчитані з потоку. При успішному читанні функція fgets() повертає вказівник на прочитаний рядок, а в разі помилки – повертає NULL.

Дія функції така: з потоку fp у ділянку оперативної пам’яті, на яку вказує str, зчитується послідовність символів до символу нового рядка чи символу кінця файлу, але не більше, ніж max-1 символів. Якщо в процесі введення зчитується символ ‘\n’, то він заноситься в str, за ним записується ‘\0’ і процес введення припиняється. У противному разі в str послідовно заноситься max-1 символів потоку, за якими записується нуль-

символ. Зауважимо також, що в разі зчитування з файлу цілого рядка або його кінцевої частини, в str перед ‘\0’ буде записаний символ ‘\n’, якщо потік відкрито в текстовому режимі, або пару символів “\r\n”, якщо потік відкрито як бінарний.

Запис заданого рядка str у потік fp виконує функція

char * fputs(char * str, FILE * fp);

яка повертає ненульове значення за умови успішного виконання та EOF в разі невдачі.

Запис даних у файл за списком форматних специфікацій здійснює функція

int fscanf(FILE * fp, <рядок_форматів>, <список_вказівників>);

яка є аналогом функції scanf().

Форматне виведення даних з потоку виконує функція

int fprintf(FILE * fp, <рядок_форматів>, <список_виразів>);

Розглянемо функції, які стосуються поточної позиції файлу. Кожна операція читання або запису в файл пов’язана зі зміщенням внутрішнього вказівника (індикатора) поточної позиції файлу на відповідну кількість байтів. У багатьох задачах потрібно керувати цим індикатором.

Основною функцією позиціювання потоку даних є функція

int fseek(FILE * fp, long offset, int base);

Функція fseek() переміщує вказівник поточної позиції файлу, пов’язаного з потоком fp, на кількість байтів, заданих параметром offset. Якщо offset>0, то вказівник поточної позиції зсувається в напрямку до кінця файлу, якщо ж offset<0, то вказівник зсувається в напрямку до початку файлу. Параметр base задає базис, відносно якого здійснюється переміщення поточної позиції. Він може бути одною із трьох макроконстант:

- SEEK_SET – за базис береться початок файлу;
- SEEK_CUR – за базис береться поточна позиція файлу;
- SEEK_END – за базис береться кінець файлу.

При успішному виконанні fseek() повертає 0, а при виникненні помилки – ненульове значення.

Функція

void rewind(FILE * fp);

встановлює вказівник поточної позиції відкритого файлу на початок цього файлу.

Значення вказівника поточної позиції файлу можна отримати через функцію

long ftell(FILE * fp);

Функція повертає кількість байтів від початку файлу до поточної позиції вказівника. Для текстових файлів це значення може бути неточним через перетворення символів кінця рядка.

Зберегти значення вказівника поточної позиції файлу можна також за допомогою функції

int fgetpos(FILE * fp, long * fpos);

Це значення запам’ятовується в змінній, що адресується вказівником fpos. При успішному завершенні функція повертає 0, а при невдачі – ненульове значення.

Для відновлення значення поточної позиції файлу, збереженого раніше через звертання до функції fgetpos(), призначена функція

int fsetpos(FILE * fp, long * fpos);

Функція

int feof(FILE * fp);

перевіряє, чи досягнуто кінця файлу, пов’язаного з потоком fp. Повертає нуль, якщо не виявлена ознака кінця файлу, інакше – ненульове значення. Всі операції читання з файлу після досягнення його кінця вважаються помилковими. Для аналізу помилок файлового введення/виведення можна використовувати функції ferorr(), perorr(), clearer().

Високорівневий потокоорієтований файловий обмін даними використовує проміжну буферизацію. Для кожного відкритого файлу в оперативній пам’яті створюється буфер обміну заданої ємності. Зазвичай, ємність буфера для файлів на дисках кратна ємності

сектора диску. Якщо файл відкривається для читання, то буфер (буфер введення) відразу заповнюється початковою порцією байтів цього файлу. Коли всі дані з буфера зчитано, автоматично у буфер заноситься нова порція байтів. Якщо ж здійснюється запис даних у файл, то вони спочатку заносяться у буфер (буфер виведення), а вже звідти переписуються у файл за кожної з умов: заповнено весь буфер, виконується очищення буфера, відбувається закриття файлу, програма завершує роботу. Буферизація забезпечує швидкодію обміну даними.

Очищення/переписування буфера потоку fp виконує функція

int fflush(FILE * fp);

При успішному завершення функція повертає 0, а при виникненні помилки – макроконстанту EOF. Для роботи з буферами можна використовувати функції setbuf(), setvbuf().

Приклад 1

Створити текстовий файл. Вивести його на екран та підрахувати у ньому кількість ком.

Аналіз задачі

Створювати текстовий файл можна записуючи у нього символи або рядки. Доцільно створити окрему функцію, яка буде створювати файл. Ми будемо записувати у файл рядки до тих пір, поки не буде введений спеціальний рядок.. Для перевірки правильності створеного файлу розробимо функцію виведення вмісту файлу на екран. Нарешті, створимо функцію обробки файлу. У ній послідовно читатимемо файл від початку до кінця і порівнюватимемо символи файлу з комою. Якщо зустрілася кома, то лічильник збільшуємо на 1. Перед читанням файлу початкове значення лічильника рівне 0.

Зауважимо, що текстовий файл можна створювати не тільки програмно, а й у «Блокноті».

Алгоритм розв’язування задачі полягає у послідовному виклику описаних функцій.

```
#include "stdafx.h"
#include "windows.h"
#include "string.h"
// Функція створює текстовий файл, пов’язаний з потоком p. У файл послідовно
// заносяться рядки тексту.
void StvorenFile(FILE *p);
// Функція виводить на екран текстовий файл, пов’язаний з потоком p.
void DrucFile(FILE *p);
// Функція призначена для підрахунку кількості ком в текстовому файлі, пов’язаному з
// потоком p. Знайдену кількість ком функція повертає в точку виклику.
int ObrobkaFile(FILE *p);

int _tmain(int argc, _TCHAR* argv[])
{       int k;
        char filename[120];                 // Змінна для імені файлу
        printf ("Vvedit imja faylu\n");
        gets(filename);                     // Введення імені файлу з клавіатури
        FILE * fp;                          // Оголошення вказівника на потік
        fp=fopen(filename, "w");            // Відкриття потоку (файлу) для запису
        if (fp==NULL)                       // Перевірка коректності відкриття файлу
        {       puts("file ne vidkryvsja");
                exit(0);
        }
        StvorenFile(fp);                    // Створюється файл
```

```
	fp=fopen(filename, "r");		// Відкриття потоку для читання
	if (fp==NULL)
	{	puts("file ne vidkryvsja");
		exit(0);
	}
	printf ("Druk file\n");
	DrucFile(fp);				// Виведення вмісту файлу на екран
	rewind(fp);			// Встановлення поточної позиції файлу на його початок
	k=ObrobkaFile(fp);		// Підрахунок числа ком в тексті
	printf ("Kilkist kom u teksti = %d\n", k);
	system("pause");
	return 0;
}
//-----------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{	printf ("Vvedit rjadky u file. Osnanniy rjadok - enter\n");
	char s[128];				// Оголошення рядка s
	fgets(s, 127, stdin);		// Читання з потоку stdin рядка і запам'ятовування
						його в s (введення рядка з клавіатури)
// Цикл для створення файлу. Цикл припиняє роботу, коли буде введений «порожній»
// рядок, тобто курсор на екрані стоїть на початку нового рядка і натискається клавіша
// Enter. Тоді в stdin запишеться рядок "\n".
	while (strcmp(s, "\n")!=0)
	{	fputs(s, p);			// Запис рядка s в потік p
		fgets(s, 127, stdin);		// Знову вводимо рядок s з клавіатури
	}
	fclose(p);			// Закриваємо потік. Створюється ознака кінця файлу
}
//-----------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{	char s[128];				// Оголошення рядка s
	fgets(s, 127, p);			// Читаємо з потоку (файла) p рядок і
							запам'ятовуємо його в s
// Цикл для перегляду файлу. Цикл припиняє роботу, коли буде досягнутий кінець файлу
	while (!feof(p))
	{	fputs(s, stdout);		// Виведення рядка s на екран
		fgets(s, 127, p);		// Знову читаємо рядок з потоку p
	}
}
//-----------------------------------------------------------------------------------------------------------
int ObrobkaFile(FILE *p)
{	int i, n, kilk=0;			// Лічильник kilk для підрахунку ком
	char s[128];				// Оголошення рядка s
	fgets(s, 127, p);			// Читаємо з потоку (файла) p рядок і
							запам'ятовуємо його в s
// Цикл для перегляду файлу. Цикл припиняє роботу, коли буде досягнутий кінець файлу
	while (!feof(p))
	{	n=strlen(s);			// Визначення довжини рядка s
		for (i=0; i<n; i++)		// Цикл для послідовного перегляду рядка s
			if (s[i]==',')		// Символ рядка є комою
				kilk++;		// Збільшуємо лічильник
		fgets(s, 127, p);		// Знову читаємо рядок з потоку p
```

```
    }
    return kilk;
}
```

```
Uvedit imja faylu
D:\\text.txt
Uvedit rjadky u file. Osnanniy rjadok - enter
За вікном зима, падає сніг, холодно.
Раз, два, три, чотири, пять.
Я іду шукати.

Druk file
За вікном зима, падає сніг, холодно.
Раз, два, три, чотири, пять.
Я іду шукати.
Kilkist kom u teksti = 6
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 2

Дано текстовий файл f цілих чисел. Знайти кількість найменших і найбільших цілих чисел у цьому файлі. Знайти число, яке є середнім найменшого і найбільшого чисел. Використовуючи допоміжний файл h, переписати числа файлу f у файл g так, щоб спочатку були записані всі числа менші середнього, а потім усі числа більші або рівні середньому.

Аналіз задачі

Розв'язування задачі складається з наступних кроків:

1. Створити текстовий файл **f** цілих чисел.
2. За перший перегляд файлу **f** знайти найменше і найбільше числа та підрахувати їх кількість.
3. Знайти середнє найменшого і найбільшого чисел.
4. Встановити поточний вказівник файлу **f** на його початок. За другий перегляд файлу **f** переписати числа файлу **f** у файл **g** за вказаним принципом.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
// Функція створює текстовий файл f, пов'язаний з потоком p. У файл заносяться цілі
// числа. Якщо введене з клавіатури число 999, то створення файлу завершується, а це
// число не записується у файл.
void StvorenFile(FILE *p);
// Функція виводить на екран текстовий файл цілих чисел, пов'язаний з потоком p.
void DrucFile(FILE *p);
// Функція здійснює пошук у файлі f, пов'язаного з потоком p, найбільшого і найменшого
// цілих чисел та підраховує їх кількість. Через вказівник pmax із функції у програму буде
// передане найбільше число, через вказівник pmin – найменше число, через pkmax –
// кількість найбільших чисел, через pkmin – кількість найменших чисел у файлі f.
void PoshukMimMax(FILE *p, int *pmax, int *pmin, int *pkmax, int *pkmin);
// Функція створює новий текстовий файл g, пов'язаний з потоком g. У функцію через
// вказівник p передається файл f. Параметр s служить для передачі у функцію середнього
// найменшого і найбільшого чисел.
void StvorenFileG(FILE *p, FILE *g, double s);

int _tmain(int argc, _TCHAR* argv[])
```

```
{       setlocale (LC_ALL,"RUS");
        int max, min;                   // Оголошення змінних для найбільшого і найменшого
                                                цілих чисел відповідно
        int kilkmax, kilkmin;           // Змінна kilkmax призначена для підрахунку кількості
                                                найбільших чисел,
                                                а kilkmin – кількості найменших чисел
        char filenamef[120];            // Змінна для імені файлу f
        char filenameg[120];            // Змінна для імені файлу g
        FILE * fpf;             // Оголошення вказівника на потік, пов’язаного з файлом f
        FILE * fpg;             // Оголошення вказівника на потік, пов’язаного з файлом g
        printf ("Введіть імя файлу f\n");
        gets(filenamef);
        fpf=fopen(filenamef, "w");              // Відкриття файлу f для запису
        if (fpf==NULL)                          // Перевірка коректності відкриття файлу f
        {       puts("Файл f не відкритий\n");
                exit(0);
        }
        StvorenFile(fpf);                       // Створюється файл f
        fpf=fopen(filenamef, "r");              // Відкриття файлу f для читання
        if (fpf==NULL)                          // Перевірка коректності відкриття файлу f
        {       puts("Файл f не відкритий\n");
                exit(0);
        }
        printf ("Друк файлу f\n");
        DrucFile(fpf);                          // Виведення вмісту файлу f на екран
        rewind(fpf);            // Встановлення поточної позиції файлу f на його початок
        PoshukMimMax(fpf, &max, &min, &kilkmax, &kilkmin);          // Пошук у файлі f на
                        найбільшого і найменшого чисел та підрахунок їх кількостей
        printf ("Найменше число = %d.  Іх кількість = %d\n", min, kilkmin);
        printf ("Найбільше число = %d.  Іх кількість = %d\n", max, kilkmax);
        double s;
        s=(max + min)/2.;                       // Обчислення середнього числа
        printf ("Середне число = %lf\n", s);
        rewind(fpf);            // Встановлення поточної позиції файлу f на його початок
        printf ("Введіть імя файлу g\n");
        fflush(stdin);                  // Очищення стандартного потоку введення stdin
        gets(filenameg);                        // Введення імені файлу g
        fpg=fopen(filenameg, "w");              // Відкриття файлу g для запису
        if (fpg==NULL)                          // Перевірка коректності відкриття файлу g
        {       puts("Файл g не відкритий\n");
                exit(0);
        }
        StvorenFileG(fpf, fpg, s);              // Створення файлу g
        fclose(fpf);                            // Закриття файлу f
        fpg=fopen(filenameg, "r");              // Відкриття файлу g для читання
        if (fpg==NULL)                          // Перевірка коректності відкриття файлу g
        {       puts("Файл g не відкритий\n");
                exit(0);
        }
        printf ("Друк файлу g\n");
        DrucFile(fpg);                          // Виведення вмісту файлу g на екран
        fclose(fpg);                            // Закриття файлу g
```

```
	system("pause");
	return 0;
}
//-----------------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{	printf ("Введіть цілі числа. Останне число - 999\n");
	int k;
	scanf("%d", &k);			// Введення числа з клавіатури
	while (k!=999)			// Цикл працює до тих пір, поки не введене число 999
	{	fprintf(p, "%d ", k);		// Запис числа в файл
		scanf("%d", &k);		// Введення числа з клавіатури
	}
	fclose(p);			// Закриваємо потік. Створюється ознака кінця файлу
}
//-----------------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{	int k;
	fscanf(p, "%d", &k);			// Читаємо число з файлу
	while (!feof(p))			// Поки не кінець файлу
	{	printf ("%d  ", k);		// Виводимо число на екран
		fscanf(p, "%d", &k);		// Знову читаємо число з файлу
	}
	printf ("\n");
}
//-----------------------------------------------------------------------------------------------------------------
void PoshukMimMax(FILE *p, int *pmax, int *pmin, int *pkmax, int *pkmin)
{	int k;
	fscanf(p, "%d", &k);		// Читаємо перше число з файлу
	*pmax=k;			// Початкове значення найбільшого числа
	*pmin=k;			// Початкове значення найменшого числа
	*pkmax=1;			// Кількість найбільших чисел
	*pkmin=1;			// Кількість найменших чисел
	fscanf(p, "%d", &k);		// Читаємо наступне число з файлу
	while (!feof(p))		// Поки не кінець файлу
	{	if (*pmax < k)		// Знайшли ще більше число ніж *pmax
		{	*pmax=k;	// Нове найбільше число
			*pkmax=1;	// Кількість найбільших чисел
		}
			else
				if (*pmax == k)		// Знайдене ще одне найбільше число
					(*pkmax)++;	// Збільшуємо кількість найбільших чисел
		if (*pmin > k)		// Знайшли ще менше число ніж *pmin
		{	*pmin=k;	// Нове найменше число
			*pkmin=1;	// Кількість найменших чисел
		}
			else
				if (*pmin == k)		// Знайдене ще одне найменше число
					(*pkmin)++;	// Збільшуємо кількість найменших чисел
		fscanf(p, "%d", &k);			// Читаємо наступне число з файлу
	}
}
//-----------------------------------------------------------------------------------------------------------------
```

```
void StvorenFileG(FILE *p, FILE *g, double s)
{       FILE *h;                                // Оголошення вказівника на потік,
                                                 пов'язаного з допоміжним файлом h
        h=fopen("D:\\hhh", "w");        // Відкриття файлу h для запису
        if (h==NULL)                    // Перевірка коректності відкриття файлу h
        {       puts("Файл h не відкритий\n");
                exit(0);
        }
        int k;
        fscanf(p, "%d", &k);            // Читаємо перше число з файлу f
        while (!feof(p))                // Поки не кінець файлу f
        {       if (k < s)              // Якщо число менше за середнє
                        fprintf(g, "%d ", k);           // Записуємо його в файл g
                        else
                                fprintf(h, "%d ", k);   // Інакше, записуємо його в файл h
                fscanf(p, "%d", &k);            // Читаємо чергове число з файлу f
        }
        fclose(h);                      // Закриваємо потік. Створюється ознака кінця файлу h
        h=fopen("D:\\hhh", "r");        // Відкриття файлу h для читання
        if (h==NULL)                    // Перевірка коректності відкриття файлу h
        {       puts("Файл h не відкритий\n");
                exit(0);
        }
        fscanf(h, "%d", &k);                    // Читаємо перше число з файлу h
        while (!feof(h))                        // Поки не кінець файлу h
        {       fprintf(g, "%d ", k);           // Записуємо число в файл g
                fscanf(h, "%d", &k);            // Читаємо чергове число з файлу h
        }
        fclose(h);                              // Закриваємо файл h
        fclose(g);                              // Закриваємо файл g
}
```

```
Введіть імя файлу f
D:\\fff
Введіть цілі числа. Останне число - 999
22 44 12 -11 10 -11
21 44 -5 -8 44 30
999
Друк файлу f
22  44  12  -11  10  -11  21  44  -5  -8  44  30
Найменше число = -11.  Іх кількість = 2
Найбільше число = 44.  Іх кількість = 3
Середне число = 16,500000
Введіть імя файлу g
D:\\ggg
Друк файлу g
12  -11  10  -11  -5  -8  22  44  21  44  44  30
Для продолжения нажмите любую клавишу . . .
```

Результати роботи програми.

Задачі

1. Задано текстовий файл, єдиний рядок якого містить 121 символ. Створити новий текстовий файл із символів першого файлу за таким алгоритмом: 121 символ переписують у вигляді матриці з 11 рядками та 11 стовпцями. Потім кожний парний рядок записують у зворотному порядку; після цього у зворотному порядку записують кожний непарний рядок.

2. Користувачеві пропонується вводити з клавіатури дані у текстовий файл. Кожен рядок файлу містить назву фірми, назву товару та ціну в доларах. Перерахувати ціни у гривні за поточним курсом та дописати до рядків отримані значення. Результати записати у новий текстовий файл.

☺ Індивідуальні завдання

Основний рівень

1. Дано текстовий файл. Надрукувати всі його рядки, що містять як фрагмент слово “риба”.
2. Знайти в даному файлі слова паліндроми і записати їх у новий файл (паліндроми - слова, що читаються однаково зліва направо і справа наліво).
3. Дано текстовий файл. Надрукувати всі його рядки, які містять більше 10 символів.
4. Дано текстовий файл. Підрахувати кількість рядків, які починаються з букв “А” або “а”.
5. Дано текстовий файл. Знайти довжину найдовшого рядка.
6. Дано текстовий файл. Обчислити кількість порожніх рядків і записати всі не порожні рядки в новий файл.
7. Дано текстовий файл. Знайти кількість входжень символу @.
8. Дано текстовий файл. Підрахувати в ньому кількість речень (речення відокремлюються крапками).
9. Дано текстовий файл. Дописати до нього наступні дані: кількість символів у кожному рядку та кількість рядків.
10. Дано текстовий файл, елементами якого є цілі числа. Знайти максимальний елемент і підрахувати кількість таких елементів
11. Дано текстовий файл. З’ясувати, чи є в ньому рядок, що починається з букви “т”. Якщо так, то визначити номер першого з таких рядків.
12. Дано текстовий файл. Підрахувати, скільки разів у ньому зустрічається слово “мама”.
13. Дано текстовий файл. Надрукувати 1-й символ 1-го рядка, 2-й символ 2-го рядка і т.д.
14. Дано текстовий файл. Переписати в інший файл ті його рядки, у яких більше 30 символів.
15. Дано текстовий файл, який містить символьні рядки. Знайти кількість рядків, що починаються та закінчуються однаковими символами.

Підвищений рівень

1. Дано текстовий файл, елементами якого є дійсні числа $a_1, a_2, a_3, ..., a_n$. Створити файл, елементами якого будуть числа $a_1, a_1+a_2, a_1+a_2+a_3, ..., a_1+a_2+...+a_n$.
2. Дано текстовий файл, елементами якого є дійсні числа $a_1, a_2, a_3, ..., a_n$. Створити файл, елементами якого будуть числа $a_1^2, a_1 \cdot a_2, a_1 \cdot a_3, a_1 \cdot a_4, ..., a_1 \cdot a_n$.
3. Дано текстовий файл, елементами якого є дійсні числа $a_1, a_2, a_3, ..., a_n$. Створити файл, елементами якого будуть числа $a_1, -a_1 \cdot a_2, a_1 \cdot a_2 \cdot a_3, -a_1 \cdot a_2 \cdot a_3 \cdot a_4, ..., (-1)^{n+1} a_1 \cdot a_2 ... a_n$.
4. Дано текстовий файл, елементами якого є дійсні числа $a_1, a_2, a_3,..., a_n$. Створити файл, у якому ці числа будуть розташовані в такому порядку $a_2, a_3, ..., a_n, a_1$.
5. Дано текстовий файл, елементами якого є дійсні числа $a_1, a_2, a_3, ..., a_n$. Створити файл дійсних чисел $b_1, b_2, b_3,..., b_n$, де $b_i = \frac{a_i}{1+(a_1+...+a_i)^2}$, i=1, 2, ..., n.
6. Дано текстовий файл, елементами якого є дійсні числа $a_1, a_2, a_3,..., a_n$. Створити файл, елементами якого будуть “середні” значення даних елементів. “Середні” значення отримують за формулою $a_i = \frac{a_{i-1}+a_i+a_{i+1}}{3}$, де i=2, 3, ..., n-1.

7. Дано текстовий файл, елементами якого є дійсні числа a_1, a_2, a_3, ..., a_n. Відомо, що $a_1>0$, та серед a_2, a_3, ..., a_n є хоча б одне від'ємне число. Отримати суму елементів, які передують першому від'ємному числу.
8. Дано текстовий файл, елементами якого є дійсні числа a_1, a_2, a_3, ..., a_n. Відомо, що $a_1>0$, та серед a_2, a_3, ..., a_n є хоча б одне від'ємне число. Створити текстовий файл, елементами якого будуть числа a_1, $a_1 \cdot a_2$, $a_1 \cdot a_2 \cdot a_3$, ..., $a_1 \cdot a_2 ... a_k$, де a_1, a_2, $a_3 ... a_k$ – елементи даного файлу до першого від'ємного числа.
9. Дано текстовий файл, елементами якого є дійсні числа a_1, a_2, a_3, ..., a_n. Створити файл, елементи якого b_1, b_2, ..., b_n отримані таким чином: $b_1=a_1$, $b_n=a_n$, $b_i = \frac{a_{i+1} - a_i}{3}$, i=2, 3, ..., n.
10. Дано текстовий файл, елементами якого є дійсні числа a_1, a_2, a_3, ..., a_n. Створити файл, у якому елементи будуть розташовані в такому порядку a_6, a_7, ..., a_n, a_1, a_2, a_3, a_4, a_5.
11. Дано текстовий файл, елементами якого є натуральні числа a_1, a_2, a_3, ..., a_n. Створити два нових файли. У перший переписати з даного файлу всі парні числа, у другий – всі непарні.
12. Дано текстовий файл, елементами якого є дійсні числа a_1, a_2, a_3, ..., a_n. Створити два нових файли. У перший переписати з даного файлу всі від'ємні числа, у другий – всі додатні.
13. Дано текстовий файл **f** цілих чисел. Отримати з **f** файл **g**, виключивши повторні входження чисел.
14. Дано текстовий файл **f** цілих чисел, що не дорівнюють нулю. Файл **f** містить однакову кількість додатних та від'ємних чисел. Використовуючи допоміжний файл **h**, переписати числа файлу **f** у файл **g** так, щоб спочатку були записані всі додатні, а потім усі від'ємні числа.
15. Дано текстовий файл **f** цілих чисел, що не дорівнюють нулю. Файл **f** містить однакову кількість додатних та від'ємних чисел. Використовуючи допоміжний файл **h**, переписати числа файлу **f** у файл **g** так, щоб не було двох сусідніх чисел з одним знаком.

♖ Додаткові задачі

1. У кожному рядку текстового файлу знайти найдовшу послідовність цифр. Значення її довжини перетворити на рядок, який дописати на початку рядка вихідного файлу. Результати записати у новий файл.
2. Створити програму генерування текстового файлу, у якому міститься картинка, що зображає множення «стовпчиком» двох заданих натуральних чисел. Можливий приклад

```
   39624
 x  8503
  118872
+ 198120
  316992
 336922872
```

5. Дано текстовий файл із речень української мови. Написати програму, що аналізує текст і виводить таблицю про число входжень кожного символу алфавіту до тексту.
6. Дано текстовий файл із речень української мови. Написати програму, що аналізує текст і виводить таблицю про число входжень слів до тексту, які складаються з одного символа, двох символів, трьох символів і т. д.
7. Дано текстовий файл із речень української мови. Написати програму, що аналізує текст і виводить таблицю про число входжень кожного слова до тексту.

☝ Питання для самоконтролю

1. Що таке файл? Що розуміють під фізичним та логічним файлами?
2. Що спільного між масивом та файлом? Чи існують між ними відмінності?
3. За якими ознаками можна класифікувати файли?
4. З якими видами файлів можна працювати в мові С?
5. Що розуміють під потоком у мові С?
6. Які дії можна виконувати з потоками?
7. З яких етапів складається робота з файлами?
8. Що таке вказівник поточної позиції файлу?
9. Синтаксис оголошення вказівника на потік.
10. З яких компонентів складається типу FILE? Де оголошений цей тип?
11. Вкажіть прототип функції відкриття файлу. Які її параметри і що вона повертає? У яких режимах можна відкривати файл?
12. Вкажіть прототип функції закриття потоку.
13. Назвіть стандартні потоки та їх призначення.
14. Вкажіть прототипи функцій для введення/виведення символів у файли. Охарактеризуйте їхню роботу.
15. Вкажіть прототипи функцій для введення/виведення рядків у файли. Охарактеризуйте їхню роботу.
16. Вкажіть прототип функції для введення/виведення форматних даних у файли. Охарактеризуйте їхню роботу.
17. Вкажіть прототипи функцій для переміщення вказівника поточної позиції файлу. Охарактеризуйте її параметри і роботу.
18. Які функції працюють з вказівником поточної позиції файлу?
19. Для чого потрібна буферизація обміну даними?
20. Вкажіть функції для роботи з буферами.

Практичне заняття № 11

*Тема: **Двійкові файли***

Студент повинен знати: поняття фізичного та логічного файлів, типи файлів і оголошення вказівників на потік, встановлення відповідності між фізичним і логічним файлами, стандартні функції для роботи з файлами, зчитування і запис файлів прямого доступу.

Теоретичні відомості

Теоретичний матеріал цього заняття в основному викладений у попердній темі. Тому доповнимо його інформацією про двійкові (бінарні) файли.

У мові С двійкові файли є файлами прямого доступу і вони трактуються як послідовності байтів. Компоненти бінарного файлу нумеруються починаючи з нуля. Між компонентами не записуються жодні роздільники. Такого поняття, як рядок, а отже, і маркер кінця рядка, для бінарних файлів не існує.

Компоненти двійкового файлу повинні бути одного типу, розмір яких фіксований. Найпростіший варіант компоненти – це байт. Однак компонентами можуть бути числа, структури та інші дані. Функція fseek() дозволяє встановлювати вказівник поточної позиції файлу на необхідний компонент (позицію), а потім можна виконувати операції читання чи запису. Отже двійкові файли можна змінювати.

Основними функціями для читання чи запису у двійкові файли є fread() та fwrite(). Вони дозволяють здійснювати обмін даними блоками. Ці функції також можна застосовувати до текстових файлів.

Функція

size_t fread(void * buf, size_t size, size_t n, FILE * fp);

зчитує з файлу, пов’язаного з потоком fp, блок даних заданого розміру. Дані заносяться у ділянку пам’яті, що адресується параметром buf. Параметр size задає розмір елемента файлу в байтах, а параметр n – кількість зчитаних елементів. Тип size_t оголошений в stdio.h. Функція повертає кількість реально зчитаних елементів (не байтів). Якщо ця величина не дорівнює значенню n, то це означає, що досягнуто кінця файлу або зафіксовано помилку читання.

Запис блоку даних у потік виконує функція

size_t fwrite(void * buf, size_t size, size_t n, FILE * fp);

Її параметри такі ж самі, що й у попередньої фукції.

Приклад 1

Створити файл структур із полями: прізвище, ім’я, по-батькові, адреса та рік народження. Вивести інформацію про тих, хто народився в 1985 році.

Аналіз задачі

Створення файлу структур організуємо у вигляді функції. У файл будемо заносити структури до тих пір, поки не зустрінеться запис, у якому прізвище рівне ’#’. Також створимо функцію для виведення структур файлу на екран. Основною буде функція пошуку у файлі людей, що народилися в 1985 році. Переглядатимемо послідовно структури файлу від початку до кінця. Якщо знайдемо запис про людину, що народилася у 1985 році, то виведемо його на екран та присвоїмо змінній **l** значення 1. Перед переглядом файлу ця змінна рівна 0. Вона використовується у ролі прапорця, та інформує про наявність чи відсутність у файлі потрібних структур. Якщо після перегляду файлу змінна **l** залишиться рівною 0, то у файлі немає структур про людей із роком народження 1985.

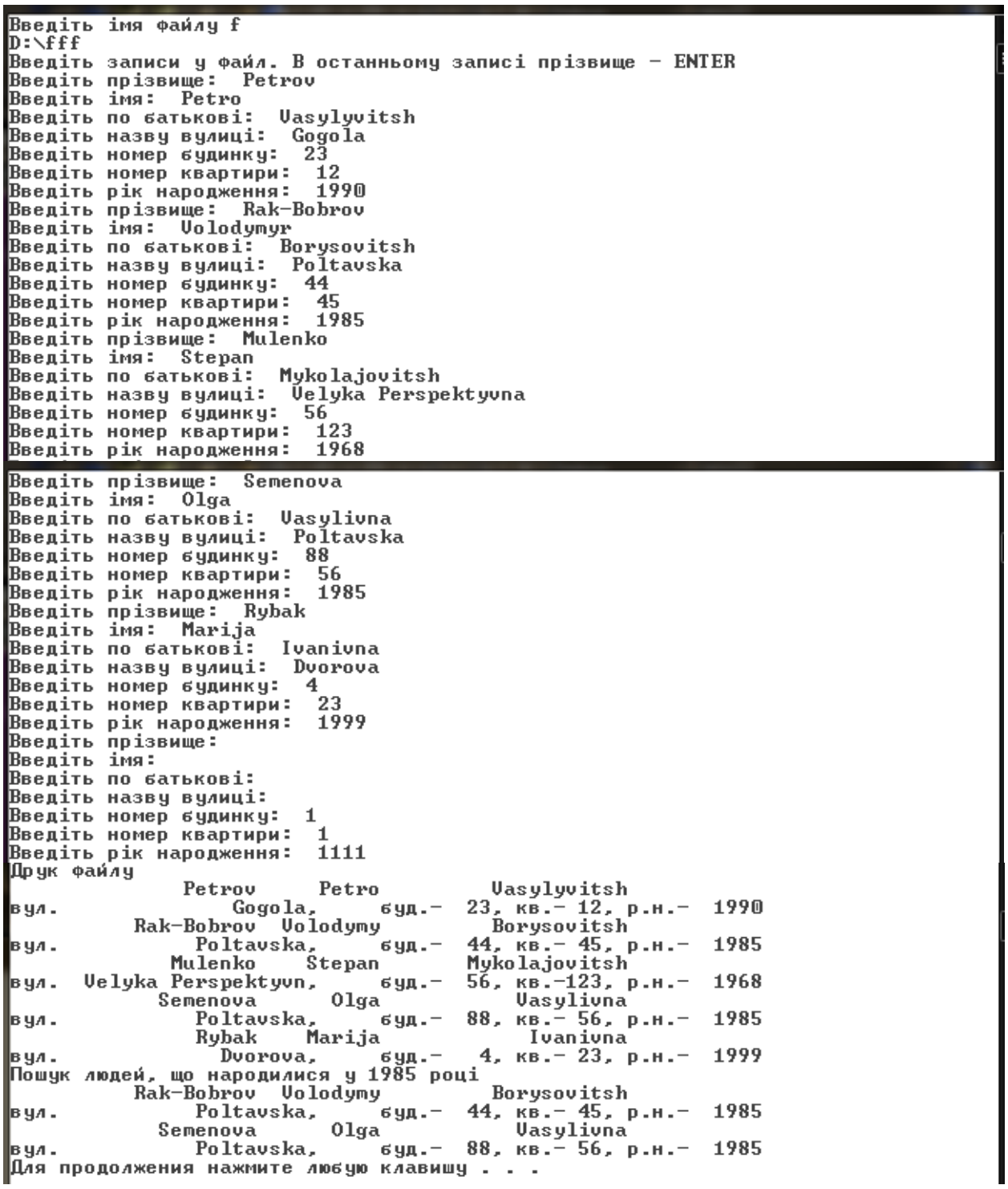
Алгоритм полягає у послідовному виклику вказаних підпрограм.

```
#include "stdafx.h"
```

```
#include "iostream"
#include "string.h"
// Визначення структури «адреса»
typedef struct adr
	{	char vulica[20];		// Назва вулиці
		int nomerbud;			// Номер будинку
		int nomerkvart;			// Номер квартири
	}	adresa;
// Визначення структури для файлу
typedef struct inf
	{	char prizv[20];			// Прізвище
		char imja[10];			// Ім'я
		char batko[20];			// По-батькові
		adresa prz;			// Адреса (вкладення структури)
		int rik;			// Рік народження
	}	anketa;
// Функція призначена для заповнення полів структури anketa. Параметром функції є
// вказівник на цю структуру, через який інформація передається із функції у програму.
void StvorenStruct(anketa *p);
// Функція замінює останній символ '\n' кожного рядка на '\0'. Це пов'язано з особливістю
// читання рядків функцією fgets(). Якщо такої заміни не робити, то при виведенні кожного
// рядка курсор буде переходити на новий рядок.
void PeretvorStruct(anketa *p);
// Функція створює файл структур.
void StvorenFile(FILE *p);
// Функція виводить вміст файлу на екран.
void DrucFile(FILE *p);
// Функція здійснює пошук у файлі людей, що народилися у 1985 році та виводить про них
// інформацію на екран. Якщо таких людей немає, то функція повертає 0, інакше - 1.
int Poshuk1985(FILE *p);

int _tmain(int argc, _TCHAR* argv[])
{	setlocale (LC_ALL,"RUS");		// Встановлення локалі
	int l;				// Оголошення змінної l, яка вказана в аналізі задачі
	char filename[120];		// Змінна для імені файлу
	printf ("Введіть ім'я файлу\n");
	gets(filename);			// Введення імені файлу з клавіатури
	FILE * fp;			// Вказівник на потік
	fp=fopen(filename, "w");	// Відкриття файлу для запису (текстовий файл)
	if (fp==NULL)
	{	puts("Файл не відкритий\n");
		exit(0);
	}
	StvorenFile(fp);		// Створення файлу
	fp=fopen(filename, "r");	// Відкриття файлу для читання
	if (fp==NULL)
	{	puts("Файл не відкритий\n");
		exit(0);
	}
	printf ("Друк файлу\n");
	DrucFile(fp);			// Виведення вмісту файлу на екран
	rewind(fp);			// Встановлення поточного вказівника на початок файлу
```

```
	printf ("Пошук людей, що народилися у 1985 році\n");
	l = Poshuk1985(fp);		// Пошук людей, що народилися в 1985 році
	if (l==0)
		printf ("Людей, що народилися у 1985 році, у файлі немає\n");
	system("pause");
	return 0;
}
//-----------------------------------------------------------------------------------------------------------
void StvorenStruct(anketa *p)
{	// Заповнення полів структури
	printf ("Введіть прізвище:  ");
	fgets(p->prizv, 20, stdin);		// Введення прізвища з клавіатури. Функція
					fgets() читає рядок до ENTER, записує в кінець
					рядка символ '\n', а потім символ '\0'. При
					виведенні таких рядків на екран курсор завжди
					буде переходити на новий рядок.
	printf ("Введіть імя:  ");
	fgets(p->imja, 10, stdin);			// Введення імені з клавіатури.
	printf ("Введіть по-батькові:  ");
	fgets(p->batko, 20, stdin);			// Введення по-батькові з клавіатури.
	printf ("Введіть назву вулиці:  ");
	fgets(p->prz.vulica, 20, stdin);		// Введення назви вулиці з клавіатури.
	printf ("Введіть номер будинку:  ");
	scanf("%d", &(p->prz.nomerbud));		// Введення номера будинку.
	printf ("Введіть номер квартири:  ");
	scanf("%d", &(p->prz.nomerkvart));		// Введення номера квартири.
	printf ("Введіть рік народження:  ");
	scanf("%d", &(p->rik));				// Введення року народження.
	fflush(stdin);  // Очищення стандартного потоку введення, бо в потоці міститься
			натиснення клавіші ENTER. Якщо потік не очистити, то
			функція fgets() прочитає рядок, який буде рівним "\n".
}
//-----------------------------------------------------------------------------------------------------------
// При заповненні полів структури функцією StvorenStruct() кожний рядок в кінці містить
// символ '\n'. Функція PeretvorStruct() замінює цей символ на '\0'. При виведенні
// перетворених рядків на екран курсор не буде переходити на початок нового рядка.
void PeretvorStruct(anketa *p)
{	int k;
	k=strlen(p->prizv);		// Визначення довжини прізвища
	p->prizv[k-1]='\0';		// Замінюємо '\n' на '\0'
	k=strlen(p->imja);		// Визначення довжини імені
	p->imja[k-1]='\0';		// Замінюємо '\n' на '\0'
	k=strlen(p->batko);		// Визначення довжини по-батькові
	p->batko[k-1]='\0';		// Замінюємо '\n' на '\0'
	k=strlen(p->prz.vulica);	// Визначення довжини вулиці
	p->prz.vulica[k-1]='\0';	// Замінюємо '\n' на '\0'
}
//-----------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{	printf ("Введіть записи у файл. В останньому записі прізвище - ENTER\n");
	anketa x;
	StvorenStruct(&x);			// Заповнення структури x
```

```
        while (strcmp(x.prizv, "\n")!=0)
        {       fwrite(&x, sizeof(anketa), 1, p);       // Запис структури у файл
                StvorenStruct(&x);                      // Заповнення структури x
        }
        fclose(p);                                      // Закриття файлу
}
//-------------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{       anketa x;
        fread(&x, sizeof(anketa), 1, p);                // Читаємо структуру x з файлу
        while (!feof(p))                                // Поки не кінець файлу
        {       PeretvorStruct(&x);                     // Перетворюємо рядки цієї структури
                // Виводимо структуру на екран
                printf ("%20s%10s%20s\n", x.prizv, x.imja, x.batko);
                printf ("вул.%20s,     буд.- %3d, кв.-%3d, р.н.-%6d\n", x.prz.vulica,
                                                x.prz.nomerbud, x.prz.nomerkvart, x.rik);
                fread(&x, sizeof(anketa), 1, p);        // Читаємо наступну структуру x з файлу
        }
}
//-------------------------------------------------------------------------------------------------------------
int Poshuk1985(FILE *p)
{       int l=0;
        anketa x;
        fread(&x, sizeof(anketa), 1, p);        // Читаємо структуру x з файлу
        while (!feof(p))                        // Поки не кінець файлу
        {       if (x.rik==1985)                // Знайшли людину, що народилася в 1985 році
                {       l=1;
                        PeretvorStruct(&x);
                        printf ("%20s%10s%20s\n", x.prizv, x.imja, x.batko);
                        printf ("вул.%20s,     буд.- %3d, кв.-%3d, р.н.-%6d\n", x.prz.vulica,
                                                x.prz.nomerbud, x.prz.nomerkvart, x.rik);
                }
                fread(&x, sizeof(anketa), 1, p);        // Читаємо структуру x з файлу
        }
        fclose(p);                              // Закриття файлу
        return l;
}
```



Результат роботи програми.

Приклад 2

Створити два файли цілих чисел. Відсортувати їх вміст. Відсортовані файли об’єднати в один відсортований файл.

Аналіз задачі

Створення та сортування файлів організуємо через відповідні функції. Об’єднання відсортованих файлів в один здійснимо методом злиття. Одночасно паралельно переглядаємо елементи двох файлів. Нехай з першого файлу читаємо число **x**, а з другого – **y**. Якщо **x**<=**y**, то записуємо в результуючий файл **x** і читаємо наступний елемент з першого файлу і знову порівнюємо його з **y**. Менше з двох чисел знову записуємо в результуючий файл. Порівняння завершується, якщо досягнутий кінець одного з файлів.

Тоді дописуємо в результуючий файл решту файлу, перегляд якого не був завершений. Алгоритм злиття реалізуємо у вигляді функції.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
// Функція створює двійковий файл цілих чисел.
int StvorenFile(FILE *p);
// Функція виводить на екран вміст двійкового файлу цілих чисел.
void DrucFile(FILE *p);
// Функція упорядковує двійковий файл цілих чисел за зростанням, який пов'язаний з
// потоком p. Через параметр n у функцію передається кількість цілих чисел у файлі.
void SortFile(FILE *p, int n);
// Функція має три параметри. Через вказівник f у функцію передається інформація про
// перший відсортований за зростанням файл цілих чисел, а через вказівник g – про другий
// аналогічний відсортований файл. Через вказівник h із функції у програму передається
// відсортований за зростанням файл цілих чисел, який є об'єднанням перших двох файлів.
void ZlyttaFile(FILE *f, FILE *g, FILE *h);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int kf, kg;                     // Змінні для підрахунку цілих чисел у файлах
        char filenamef[120];            // Змінна для імені файлу f
        char filenameg[120];            // Змінна для імені файлу g
        char filenameh[120];            // Змінна для імені файлу h
        printf ("Введіть ім'я файлу f\n");
        gets(filenamef);                // Введення з клавіатури імені файлу f
        FILE * fp;
        fp=fopen(filenamef, "wb");      // Відкриття двійкового файлу f для запису
        if (fp==NULL)
        {       puts("Файл f не відкритий\n");
                exit(0);
        }
        kf=StvorenFile(fp);             // Створення двійкового файлу f. Змінна kf
                                                містить кількість цілих чисел у файлі f.
        fp=fopen(filenamef, "r+b");  // Відкриття двійкового файлу f для читання і запису
        if (fp==NULL)
        {       puts("Файл f не відкритий\n");
                exit(0);
        }
        printf ("Друк файлу f\n");
        DrucFile(fp);                   // Виведення вмісту файлу f на екран
        fseek(fp, 0, SEEK_SET);         // Встановлення поточного вказівника на початок
                                                файлу f
        SortFile(fp, kf);               // Сортування файлу f за зростанням
        fseek(fp, 0, SEEK_SET);         // Встановлення поточного вказівника на початок
                                                файлу f
        printf ("Друк відсортованого файлу f\n");
        DrucFile(fp);                   // Виведення вмісту відсортованого файлу f на екран
        fseek(fp, 0, SEEK_SET);         // Встановлення поточного вказівника на початок
                                                файлу f
        printf ("Введіть ім'я файлу g\n");
```

```
	fflush(stdin);			// Очищення буфера стандартного потоку введення
					перед викликом функції gets()
	gets(filenameg);		// Введення з клавіатури імені файлу g
	FILE * gp;
	gp=fopen(filenameg, "wb");	// Відкриття двійкового файлу g для запису
	if (gp==NULL)
	{	puts("Файл g не відкритий\n");
		exit(0);
	}
	kg=StvorenFile(gp);		// Створення двійкового файлу g. Змінна kg
						містить кількість цілих чисел у файлі g
	gp=fopen(filenameg, "r+b"); // Відкриття двійкового файлу g для читання і запису
	if (gp==NULL)
	{	puts("Файл g не відкритий\n");
		exit(0);
	}
	printf ("Друк файлу g\n");
	DrucFile(gp);			// Виведення вмісту двійкового файлу g на екран
	fseek(gp, 0, SEEK_SET);		// Встановлення поточного вказівника на початок
						файлу g
	SortFile(gp, kg);		// Сортування файлу g за зростанням
	fseek(gp, 0, SEEK_SET);		// Встановлення поточного вказівника на початок
						файлу g
	printf ("Друк відсортованого файлу g\n");
	DrucFile(gp);			// Виведення вмісту відсортованого файлу g на екран
	fseek(gp, 0, SEEK_SET);		// Встановлення поточного вказівника на початок
						файлу
	printf ("Введіть імя файлу h\n");
	fflush(stdin);			// Очищення буфера стандартного потоку введення
					перед викликом функції gets()
	gets(filenameh);		// Введення з клавіатури імені файлу h
	FILE * hp;
	hp=fopen(filenameh, "wb");	// Відкриття двійкового файлу h для запису
	if (hp==NULL)
	{	puts("Файл h не відкритий\n");
		exit(0);
	}
	ZlyttaFile(fp, gp, hp);		// Об’єднання файлів f і g у відсортований файл h
	hp=fopen(filenameh, "rb");	// Відкриття двійкового файлу h для читання
	if (hp==NULL)
	{	puts("Файл h не відкритий\n");
		exit(0);
	}
	printf ("Друк обеднаного відсортованого файлу h\n");
	DrucFile(hp);			// Виведення файлу h на екран
	fclose(fp);			// Закриття файлу f
	fclose(gp);			// Закриття файлу g
	fclose(hp);			// Закриття файлу h
	system("pause");
	return 0;
}
```

```
//-----------------------------------------------------------------------------------------------------------------
int StvorenFile(FILE *p)
{       // Організовуємо цикл для введення цілих чисел з клавіатури і запису їх у файл,
        // що пов'язаний з вказівником p. Якщо введене число 999, то цикл припиняє
        // роботу. Це число не заноситься у файл.
        int l=0;                // Початкове значення лічильника кількості цілих чисел, які
                                        будуть записані у файл
        printf ("Введіть числа у файл. Останне число - 999\n");
        int x;
        scanf("%d", &x);        // Введення цілого числа з клавіатури
        while (x!=999)
        {       l++;            // Збільшуємо лічильник
                fwrite(&x, sizeof(int), 1, p);          // Записуємо число у файл
                scanf("%d", &x);                        // Вводимо чергове ціле число
        }
        fclose(p);                                      // Закриття файлу
        return l;                                       // Повертаємо значення лічильника
}
//-----------------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{       int x;
        fread(&x, sizeof(int), 1, p);                   // Читаємо число x з файлу
        while (!feof(p))                                // Поки не кінець файлу
        {       printf("%d   ", x);                     // Виводимо число x на екран
                fread(&x, sizeof(int), 1, p);           // Читаємо наступне число x з файлу
        }
        printf("\n");
}
//-----------------------------------------------------------------------------------------------------------------
void SortFile(FILE *p, int n)
{       // Сортування двійкового файлу цілих чисел методом бульбашок
        int i, j;
        int x, y;                       // Змінні для цілих чисел
        for(i=n-2; i>=0; i--)
                for(j=0; j<=i; j++)
                {       fseek(p, j*sizeof(int), SEEK_SET);          // Встановлюємо поточний
                                                                    вказівник файлу на j-те ціле число
                        fread(&x, sizeof(int), 1, p);   // Читаємо з файлу j-те ціле число x
                        fread(&y, sizeof(int), 1, p);   // Читаємо з файлу (j+1)-е ціле число y
                        if (x>y)                        // Переставляємо числа місцями
                        {       fseek(p, j*sizeof(int), SEEK_SET);  // Встановлюємо поточний
                                                                    вказівник файлу на j-те ціле число
                                fwrite(&y, sizeof(int), 1, p);          // Записуємо у файл число y
                                fwrite(&x, sizeof(int), 1, p);          // Записуємо у файл число x
                        }
                }
}
//-----------------------------------------------------------------------------------------------------------------
void ZlyttaFile(FILE *f, FILE *g, FILE *h)
{       // Злиття відсортованих файлів f і g в один відсортований файл h
        int x, y;                               // Змінні для цілих чисел
        fread(&x, sizeof(int), 1, f);           // Читаємо число x з файлу f
```

```
        fread(&y, sizeof(int), 1, g);           // Читаємо число y з файлу g
        while ((!feof(f))&&(!feof(g)))          // Поки не кінець обох файлів
        {       if (x<=y)
                {       fwrite(&x, sizeof(int), 1, h);  // Записуємо в файл h число y
                        fread(&x, sizeof(int), 1, f);   // Читаємо наступне число x з файлу f
                }
                        else
                        {       fwrite(&y, sizeof(int), 1, h);  // Записуємо в файл h число y
                                fread(&y, sizeof(int), 1, g);   // Читаємо число y з файлу g
                        }
        }
        // Цикл припинив роботу якщо досягнутий кінець хоч одного з файлів.
        // Файл, кінець якого не досягнутий, треба переписати в файл h
        if (!feof(f))                   // Не досягнуто кінця файлу f
                while (!feof(f))
                {       fwrite(&x, sizeof(int), 1, h);  // Записуємо в файл h число x
                        fread(&x, sizeof(int), 1, f);   // Читаємо наступне число x з файлу f
                }
        if (!feof(g))                   // Не досягнуто кінця файлу g
                while (!feof(g))
                {       fwrite(&y, sizeof(int), 1, h);  // Записуємо в файл h число y
                        fread(&y, sizeof(int), 1, g);   // Читаємо наступне число y з файлу g
                }
        fclose(h);                      // Закриваємо файл h
}
```

```
Введіть імя файлу f
D:\fff
Введіть числа у файл. Останне число - 999
5 2 4 -2 3 999
Друк файлу f
5   2   4   -2   3
Друк відсортованого файлу f
-2   2   3   4   5
Введіть імя файлу g
D:\ggg
Введіть числа у файл. Останне число - 999
8 3 -5 -1 6 9 3 4 999
Друк файлу g
8   3   -5   -1   6   9   3   4
Друк відсортованого файлу g
-5   -1   3   3   4   6   8   9
Введіть імя файлу h
D:\hhh
Друк обєднаного відсортованого файлу h
-5   -2   -1   2   3   3   3   4   4   5   6   8   9
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Задачі

1. Створити файл структур із полями: прізвище, ім’я, по батькові, посада, зарплата. Відсортувати файл за спаданням зарплати.
2. Інформація, що входить до файлу: ПІБ автора, назва книги, видавництво, вартість книги. Відсортувати файл за алфавітом прізвищ авторів, а за наявності книг одного автора додатково відсортувати за алфавітом назв книг. Вивести відсортований файл.

☺ Індивідуальні завдання

Основний рівень

1. Інформація, що входить до файлу: ПІБ, номер телефону, адреса, вартість переговорів. Вивести інформацію про клієнтів із вказаним номером телефону.
2. Скориставшись інформацією з варіанта №1, вивести інформацію про клієнтів, прізвища яких починаються на «Шев».
3. Інформація, що входить до файлу: ПІБ автора, назва книги, видавництво, вартість. Вивести інформацію про книги зазначеного автора.
4. Скориставшись інформацією з варіанта №3, обчислити загальну вартість усіх книг.
5. Скориставшись інформацією з варіанта №3, вивести інформацію про всі книги зазначеного видавництва.
6. Інформація, що входить до файлу: ПІБ, адреса, місце роботи, зарплата. Вивести інформацію про всіх співробітників КДПУ.
7. Скориставшись інформацією з варіанта №6, вивести інформацію про співробітників КДПУ, зарплата яких не перевищує 1400 грн.
8. Скориставшись інформацією з варіанта №6, вивести інформацію про співробітників із зазначеним прізвищем.
9. Інформація, що входить до файлу: розмір костюма, витрати тканини, вартість костюма, фірма. Вивести інформацію про всі костюми конкретного розміру.
10. Скориставшись інформацією з варіанта №9, вивести інформацію про вартість усіх костюмів.
11. Скориставшись інформацією з варіанта №9, вивести інформацію про костюми вартістю 1300±10 грн.
12. Скориставшись інформацією з варіанта №9, вивести інформацію про костюми, пошитих зазначеною фірмою.
13. Інформація, що входить до файлу: номер рейсу, вага багажу, кількість речей. Визначити кількість пасажирів, вага багажу яких перевищує 30 кг .
14. Скориставшись інформацією з варіанта №13, визначити середню вагу багажу.
15. Скориставшись інформацією з варіанта №13, визначити загальну кількість речей.

Підвищений рівень

1. Дано символьний файл **f**. Подвоїти в ньому усі символи, що позначають голосні звуки англійської мови.
2. Дано файл **f** натуральних чисел. Записати у файл **g** усі прості числа з файлу **f** у порядку спадання.
3. Дано символьний файл **f**. Переписати у файл **g** символи, що зустрічаються у файлі **f** двічі (рівно два рази).
4. Дано файл **f** простих чисел, записаних у порядку зростання. Вилучити з нього усі числа з парною сумою цифр.
5. Дано файл **f** натуральних чисел. Вилучити з нього усі компоненти, що є числами Фібоначчі.
6. Дано файл **f** натуральних чисел. Вилучити з нього усі компоненти, номери яких є числами Фібоначчі.
7. Дано символьний файл **f**. Вилучити повторні входження символів (залишити тільки перші).
8. Дано файл **f** натуральних чисел. Вилучити з нього числа виду 4k+1.
9. Дано символьний файл **f**. Подвоїти в ньому усі цифри, які позначають парні числа.
10. Дано символьні файли **f** та **g**. Визначити, чи дійсно вони складаються з однакових символів. Якщо вони відрізняються хоча б одним символом, то вивести символи, якими вони відрізняються.

11. Дано символьний файл **f**. Підрахувати кількість входжень у файл кожної з літер a, b, c, d, e, f та вивести результат окремими рядками.
12. Дано файл **f** натуральних чисел, кожне з яких не більше за 255. Створити множину простих чисел, на які діляться числа файлу **f.**
13. Дано файл **f** натуральних чисел. Видрукувати усі різні елементи файлу у порядку спадання.
14. Дано два файли **f** та **g** натуральних чисел, які не містять повторних входжень однакових елементів. З'ясувати, чи правда, що вони відрізняються лише порядком слідування елементів.
15. Дано файл **f** однобайтових натуральних чисел, у якому записано у порядку зростання декілька простих чисел. Дописати у файл **f** решту простих чисел, що не перевищують максимального числа з файлу **f**, не порушуючи впорядкованості елементів.

Додаткові задачі

1. Створити базу даних «Студент», робота із якою виконувалася б у діалоговому режимі. При запуску програми на екрані повинно з'являтися меню такого вигляду:
    1. Створити нову базу.
    2. Відкрити існуючу базу.
    3. Добавити запис.
    4. Вилучити запис.
    5. Сортування.
    6. Пошук.
    7. Друкування.
    8. Вихід.

   База містить записи з полями: ПІБ, стать, курс, вік, стипендія. Вказівка «Сортування» має підменю, що передбачає вибір варіантів сортування за ключами: прізвище, курс, вік. Вказівка «Пошук» теж має підменю, яке передбачає вибір варіантів пошуку за ключами: прізвище, вік, розмір стипендії.

Питання для самоконтролю

1. Яка різниця між файлами послідовного та прямого доступу?
2. Чи можна змінити зміст текстового файлу?
3. Чи можна змінити зміст двійкового файлу?
4. У чому полягає різниця між масивом та двійковим файлом?
5. Які функції призначені для роботи з поточний вказівником файлу?
6. Функція fseek(). Її призначення та параметри.
7. Функція fread(). Її призначення та параметри.
8. Функція fwrite(). Її призначення та параметри.

Практичне заняття № 12

Тема: ***Рекурсія***

Студент повинен знати: поняття рекурсії та можливості її реалізації у мові С, види рекурсій та їх особливості, механізм виконання рекурсивних програм.

Теоретичні відомості

Алгоритм називається **рекурсивним**, якщо в своїй конструкції він містить виклик самого себе. Зазвичай рекурсія реалізовується на рівні підпрограм. Підпрограма може містити виклик самої себе (рекурсивний виклик). Підпрограма з рекурсивними викликами називається рекурсивною. Процес виконання рекурсивної підпрограми складається з прямого ходу рекурсії (рекурсивне занурення), обриву (умови завершення рекурсії) та зворотного ходу (рекурсивне повернення). Величина, яка характеризує максимальну кількість незавершених рекурсивних викликів, називається глибиною рекурсії.

У мові С підтримуються два види рекурсій: пряма та непряма. При прямій рекурсії функція викликає саму себе. Непряма рекурсія – це організація взаємних викликів двох підпрограм, тобто одна підпрограма викликає другу, а друга викликає першу. Вона теж реалізується в мові через функції.

Приклад 1

Написати програму обчислення $u_n=u_{n-1}-u_{n-2}$ для даного цілого невід'ємного числа $n>1$, $u_0=0$, $u_1=1$. Створити рекурсивну функцію для обчислення u_n.

Аналіз задачі

Користуючись початковими значення членів $u_0=0$, а $u_1=1$, за співвідношення $u_n=u_{n-1}-u_{n-2}$ можна обчислити будь-який член u_n послідовності при $n>1$. Співвідношення $u_n = u_{n-1}-u_{n-2}$ підказує як побудувати рекурсивну функцію. Якщо n=0, то значення функції повинно бути рівним 0, бо $u_0=0$. Аналогічно при n=1 функція приймає значення 1. Для більших n значення функції будемо обчислювати через значення функції при n-1 та n-2. Це буде реалізовувати рекурсію.

Алгоритм полягає в обчислення функції для даного n. Зауважимо, що рекурсивний варіант обчислення послідовності є неефективним тому, що містить зайві обчислення.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
// Рекурсивна функція для обчислення un. Через параметр x у функцію передається номер
// члена послідовності. Вона повертає значення цього члена.
int Suma(int x);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int n;                          // Номер члена послідовності
        printf ("Введіть натуральне число n - ");
        scanf("%d", &n);
        printf ("%d член послідовності = %d\n", n, Suma(n));
        system("pause");
        return 0;
}
//-----------------------------------------------------------------------------------------------------------------
int Suma(int x)
{       if (x==0)                       // Обрив рекурсії
                return 0;
```

```
        else
                if (x==1)               // Обрив рекурсії
                        return 1;
                else
                        return Suma(x-1) - Suma(x-2);       // Обчислення ux = ux-1 - ux-2.
                                                            Організація рекурсивних викликів.
}
```

```
Введіть натуральне число n - 10
10 член послідовності = -1
Для продолжения нажмите любую клавишу . . . _
```

Результат роботи програми.

Приклад 2

У вхідному файлі записаний без помилок логічний вираз такого виду:
<логічний вираз>::=0|1|(<логічний вираз><операція><логічний вираз>)|
(!<логічний вираз>)
<операція>::&||
Ввести логічний вираз і обчислити його значення.

Аналіз задачі

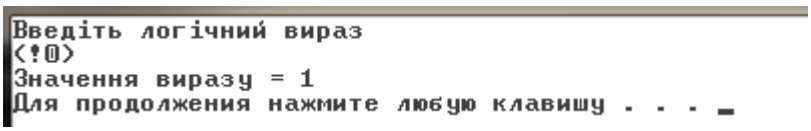
Означення логічного виразу задане у формі Бекуса-Наура і воно рекурсивне. Найпростішим логічним виразом є 0 (false) і 1 (true). Складніші логічні вирази утворюються з найпростіших за допомогою операцій ! (заперечення - not), & (кон'юнкція – and), | (диз'юнкція – or) та круглих дужок. Прикладами виразів є: 0, (!0), ((0&1)|(!(1|0))).

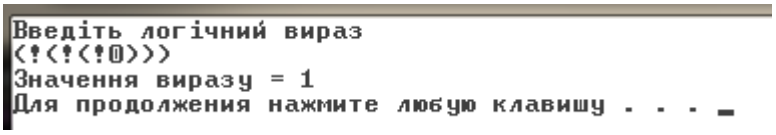
Створимо функції: Not() для реалізації операції заперечення; And() для операції кон'юнкція; Or() для операції диз'юнкція. Функція для обчислення значення логічного виразу називається Vyraz(). Вона рекурсивна і послідовно читає символи виразу, який є рядком.

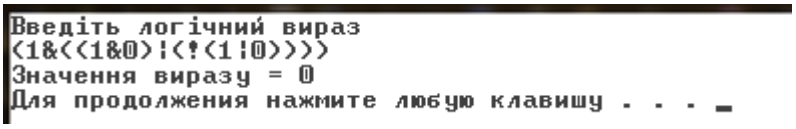
```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
// Функція реалізує операцію заперечення.
char Not(char c);
// Функція реалізує операцію кон'юнкцію.
char And(char c1, char c2);
// Функція реалізує операцію диз'юнкцію.
char Or(char c1, char c2);
// Рекурсивна функція для обчислення значення логічного виразу.
char Vyraz(void);
int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        char n;                                 // Змінна для значення логічного виразу
        printf ("Введіть логічний вираз\n");
        n=Vyraz();                              // Обчислення виразу
        printf ("Значення виразу = %c\n", n);
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------
char Not(char c)
{       // Обчислення заперечення
```

```
        if (c=='0')
                return '1';
        if (c=='1')
                return '0';
}
//-------------------------------------------------------------------------------------------------------------
char And(char c1, char c2)
{       // Обчислення кон'юнкції
        if ((c1=='1') && (c2=='1'))
                return '1';
        else
                return '0';
}
//-------------------------------------------------------------------------------------------------------------
char Or(char c1, char c2)
{       // Обчислення диз'юнкції
        if ((c1=='0') && (c2=='0'))
                return '0';
        else
                return '1';
}
//-------------------------------------------------------------------------------------------------------------
char Vyraz(void)
{       // Обчислення значення логічного виразу.
        char c, x, y, op;
        scanf("%c", &c);                        // Читаємо символ виразу (рядка)
        if ((c=='0') || (c=='1'))               // Обрив рекурсії
                return c;
        else
                if (c=='!')                     // Вираз має вигляд !V, де V – логічний вираз
                        return Not(Vyraz());            // Обчислюмо цей виразу.
                else    // Прочитаний символ є '('. Вираз може мати вигляд (!V) або (V op V)
                {       x=Vyraz();      // Обчислюємо значення виразу після '(', тобто !V або V
                        scanf("%c", &op);       // Читаємо символ після обчисленого виразу.
                        if (op==')')    // Це закрита дужка. Отже обчислювали вираз !V
                                return x;       // Повертаємо значення виразу !V
                        else    // Вираз має вигляд (V op V). Значення лівого V міститься в x
                                {       y=Vyraz();      // Обчислюємо правий V
                                        scanf("%c", &c);                // Читаємо закриту дужку
                                        if (op=='&')
                                                return And(x, y);       // Кон'юнкція
                                        if (op=='|')
                                                return Or(x, y);        // Диз'юнкція
                                }
                }
}
```

```
Введіть логічний вираз
0
Значення виразу = 0
Для продолжения нажмите любую клавишу . . .
```







Результати роботи програми.

Задачі

1. У вхідному файлі заданий текст, який закінчується крапкою. Перевірити, чи його структура задовольняє такому означенню:
   <текст>::= <елемент> | <елемент> <текст>
   <елемент>::= a | b | (<текст>) | [<текст>] | {<текст>}
2. Дано деякий текст, що закінчується крапкою (у сам текст крапка не входить). Визначити, чи є цей текст правильним записом «формули»
   <формула>::= <терм> | (<формула> <знак> <формула>)
   <знак>::= + | - | *
   <терм>::= <ім'я> | <ціле>
   <ім'я>::= <буква> | <ім'я> <буква> | <ім'я> <цифра>
   <ціле>::= <цифра> | <ціле> <цифра>
   <буква>::= a | б | в | г | д | е | ж
   <цифра>::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

☺ Індивідуальні завдання

Основний рівень

1. Підрахувати кількість золотих монет, принесених у данину пану від N+1 підданих. Перший підданий віддає одну монету, другий збільшує число монет удвічі, третій - у три рази і т.д.
2. Знайти різницю факторіалів F=m!-k!
3. Знайти НСД (найбільший спільний дільник) двох натуральних чисел за алгоритмом Евкліда.
4. Обчислити значення дробу

$$\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cfrac{1}{\ddots+\cfrac{1}{101+\cfrac{1}{103}}}}}}$$

5. Переписати довільну послідовність символів, яка вводиться з клавіатури і закінчується крапкою, у зворотному порядку.
6. Знайти n-не число Фібоначчі. (Числа Фібоначчі $u_0,u_1,u_2,\ldots$ визначаються так: $u_0=0$, $u_1=1$, $u_n=u_{n-1}+u_{n-2}$ при $n=2,3,\ldots$).

7. З клавіатури вводиться непорожня послідовність ненульових цілих чисел, яка закінчується нулем. Вивести на екран в порядку введення спочатку усі додатні числа цієї послідовності, а потім усі від'ємні.
8. Написати функцію друкування десяткових цифр цілого числа у зворотному порядку, починаючи з молодших розрядів.
9. Визначити номер та значення мінімального елемента не порожньої послідовності цілих чисел, яка закінчується нулем. Послідовність вводиться з клавіатури.
10. Знайти рекурсивно n-ну похідну f(x)=exp(a*x*x+b*x+c) для заданого x, побудувавши для $f^{(n)}(x)$ рекурентне співвідношення.
11. Обчислити число зерен, вирощених селянином за N років, якщо він посадив 10 зерен. Річний врожай складає 22 зерна на кожне посаджене зерно.
12. З клавіатури вводиться не порожня послідовність ненульових цілих чисел, яка закінчується нулем. Вивести на екран спочатку всі від'ємні числа з цієї послідовності, а потім усі додатні (у будь-якому порядку).
13. Знайти рекурсивно n-ну похідну f(x)=exp(-x*x/2) для заданого x, побудувавши для $f^{(n)}(x)$ рекурентне співвідношення.
14. Обчислити функцію y=sin(sin(sin(. . . (sin(x))))) для заданого x, у якій ім'я "sin" повторюється N раз.
15. Обчислити число рибок, вирощених в акваріумі за N років, якщо спочатку було дві рибки, а потім число рибок збільшувалося пропорційно числу років, тобто 4, 12, 48 і т.д.

Підвищений рівень

1. Дано n різних натуральних чисел. Надрукувати усі перестановки цих чисел.
2. У вхідному файлі задана не порожня послідовність дійсних чисел, яка закінчується нулем. Описати рекурсивну функцію rsum без параметрів для знаходження різниці сум додатних і від'ємних чисел цієї послідовності.
3. Створити рекурсивну підпрограму алгоритму бінарного пошуку.
4. *Швидке сортування.* Задано масив дійсних чисел. Упорядкувати їх за зростанням таким способом: вибрати довільний (наприклад, середній) елемент масиву і переставити елементи масиву так, щоб зліва від обраного елемента опинилися лише менші за нього, а справа - лише більші за нього (таким чином обраний елемент опиниться на своєму місці). Після цього застосувати той самий підхід рекурсивно до лівої та до правої частин масиву.
5. *Ханойська вежа.* Є три стержні A,B,C та n дисків різного розміру, пронумерованих від 1 до n у порядку зростання розмірів. Спочатку всі диски знаходяться на стержні A в порядку зменшення розмірів: найбільший – внизу, найменший – зверху. Потрібно перенести всі диски зі стержня A на стержень C, дотримуючись вимог: переносити можна лише по одному диску, більший диск не можна класти на менший. Стержень B допоміжний. Створити рекурсивний алгоритм, який виводить виконувані перенесення.
6. Є n населених пунктів, пронумерованих від 1 до n. Деякі пари пунктів з'єднано дорогою. Визначити, чи можна потрапити з пункту 1 до пункту n. Інформація про дороги задається у вигляді послідовності пар чисел i та j (i<j), які вказують, що i-й та j-й пункти з'єднані дорогою. Ознакою кінця цієї послідовності є пара нулів.
7. Заданий вектор x із n дійсних чисел. Описати функцію min(x) для визначення мінімального елемента вектора, використовуючи допоміжну рекурсивну функцію min1(k), яка знаходить мінімум серед останніх елементів вектора x, починаючи з к-го.
8. У вхідному файлі записана формула такого виду:
   <формула>::=<цифра>|(<формула><знак><формула>)
   <знак>::+|-|*
   <цифра>::=0|1|2|3|4|5|6|7|8|9.
   Ввести цю формулу і обчислити її значення. (Наприклад, $5 \to 5, ((2-4)*6) \to -12.$)).

9. Заданий переліковий тип ім'я=(Алла,…,Юра, ні); Нехай описані функції Батько(**x**) і Мати(**x**), значеннями яких є імена відповідно батька і матері людини з ім'ям **x** або ідентифікатор «**ні**», якщо відсутні відомості про відповідного родича. Створити логічну функцію Нащадок(**a**, **в**), яка перевіряє, чи є людина з ім'ям **в** нащадком (дитиною, онуком, правнуком, і т.д.) людини з ім'ям **a**.
10. Задана функція

```
int f(int n)
  {     if (n>100)     return n-10;
        else    return f(f(n+11))
  }
```

Обчислити f(106), f(99) і f(85).

11. Обчислити значення многочлена $y = 11\ x^{10} + 10\ x^9 + ... + 2\ x + 1$ за схемою Горнера.

12. Обчислити x^n за формулою : $x^n = \begin{cases} 1, & n = 0, \\ 1/x^{|n|}, & n < 0, \\ x \cdot x^{n-1}, & n > 0. \end{cases}$

13. Обчислити біноміальний коефіцієнт C_n^m, користуючись співвідношеннями $C_n^0 = C_n^n = 1$, $C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ при 0<m<n.
14. *Задача про 8 ферзів.* Розташувати 8 ферзів на шаховій дошці так, щоб вони не "били" один одного. Вивести всі можливі варіанти розташувань.
15. Дано послідовність дійсних чисел a_1, a_2, a_3,…, a_n . Написати рекурсивну програму створення послідовності "середніх" значень даних елементів. "Середні" значення отримують за формулою $a_i = \frac{a_{i-1} + a_i + a_{i+1}}{3}$, де i=2,3,…,n-1.

Додаткові задачі

1. Обчислити функцію Аккермана за її рекурсивним означенням:

$$A(n,m) = \begin{cases} m+1, & n = 0, \\ A(n-1,1), & n \neq 0,\ m = 0, \\ A(n-1, A(n, m-1)), & n > 0,\ m > 0. \end{cases}$$

2. У країні Тутті-Фрутті діє фінансова піраміда. Кожен із представників племені Тутті залучає до участі в піраміді трьох представників племені Тутті і чотирьох представників племені Фрутті, а кожен з представників племені Фрутті – п'ятьох представників племені Тутті і двох представників племені Фрутті. Кожен із представників племені Тутті перераховує «нагору» 5 тугриків і 30% від надходжень, отриманих «знизу», а кожен із представників племені Фрутті – 10 тугриків і 20% від надходжень, отриманих «знизу». Визначити прибуток засновника к-рівневої піраміди, якщо він є представником племені Тутті.

Питання для самоконтролю

1. Дайте рекурсивне визначення функцій. Який алгоритм називають рекурсивним?
2. Як виконується рекурсивна підпрограма?
3. Що називається прямим ходом рекурсії?
4. Що називають "глибиною" рекурсії?
5. Що називають зворотним ходом рекурсії?
6. Чи рівносильні поняття «рекурсія» та «цикл»?
7. У чому переваги та недоліки застосування рекурсивних підпрограм?
8. Як організувати непряму рекурсію в мові С?

Практичне заняття № 13

Тема: ***Сортування***

Студент повинен знати: постановку задачі сортування, основні алгоритми сортування та оцінки їхньої ефективності, застосування цих алгоритмів для розв'язування практичних задач.

Теоретичні відомості

Сортуванням або упорядкуванням списку об'єктів називається розташування цих об'єктів за зростанням або спаданням згідно певного лінійного відношення порядку. Відомо багато методів сортування, що відрізняються швидкодією та обсягом оперативної пам'яті, що при цьому використовується. Серед цих методів можна виділити методи **внутрішнього** та **зовнішнього** сортування. При внутрішньому сортуванні всі дані, що сортуються, повністю розміщуються в оперативній пам'яті комп'ютера, де можна отримати доступ до даних у будь-якому порядку. Зовнішнє сортування застосовується тоді, коли обсяг даних, що треба відсортувати, досить великий, і їх всіх не можна розмістити в оперативній пам'яті.

Будемо розглядати методи внутрішнього сортування. Їх прийнято поділяти на дві групи: елементарні (прямі) та удосконалені.

Найвідомішими елементарними методами сортування є:

- сортування вставкою (включенням);
- сортування вибором;
- сортування обміном (бульбашкове сортування).

З удосконалених методів сортування найчастіше використовуються такі:

- швидке сортування або метод Хоара;
- сортування включенням зі спадним приростом або метод Шелла;
- сортування за допомогою дерев або пірамідальне сортування;
- сортування методом злиття.

Розглянемо постановку задачі внутрішнього сортування. Будемо вважати об'єктами сортування записи, що містять одне або декілька полів. Одне з полів, що називається **ключем**, має такий тип даних, для якого визначене відношення лінійного порядку '$\le$' ('$\ge$'). Найчастіше тип ключа є числовим, символьним або рядковим. Зазвичай тип ключа може бути будь-яким, аби лише для даних цього типу можна було б визначити відношення «менше» чи «менше або рівно» («більше» чи «більше або рівно»).

Задача сортування полягає в упорядкуванні послідовності записів таким чином, щоб значення ключового поля утворювали не спадну (не зростаючу) послідовність. Іншими словами, записи $R_1, R_2, ..., R_n$ зі значеннями ключів $k_1, k_2, ..., k_n$ треба розташувати в такому порядку $R_{i1}, R_{i2}, ..., R_{in}$, щоб $k_{i1} \le k_{i2} \le ... \le k_{in}$. Не вимагається, щоб усі записи були різними. Якщо є записи з однаковими значеннями ключів, то в упорядкованій послідовності вони розташовуються поруч один з одним у будь-якому порядку.

Метод «бульбашок». Щоб описати основну ідею методу, припустимо що треба відсортувати елементи послідовності (масиву) X_1, X_2,…, X_n за зростанням. Виконаємо перший прохід послідовності наступним чином. На кожному кроці порівнюємо j-й елемент послідовності з (j+1)-им, змінюючи j від 1 до n-1. Якщо j-й елемент більший за (j+1)-ий, то переставляємо їх місцями (виконуємо обмін), інакше не виконуємо перестановки. У результаті виконання першого проходу найбільший елемент послідовності буде знаходитися на останньому (n-му) місці. Другий прохід виконується аналогічно, але він виконується для підпослідовності елементів X_1, X_2,…, X_{n-1}, тобто без останнього елемента. Аналогічно виконуються наступні проходи. При кожному

черговому проході кількість елементів підпослідовності зменшується на 1. Всього виконується n-1 проходів. Часова складність (ефективність) методу «бульбашок» дорівнює $O(n^2)$ для послідовності з n елементів. Фрагмент цього алгоритму псевдомовою має вигляд

```
for  i:=n-1 downto 1 do
     for  j:=1 to i do
          if X[j]>X[j+1] then  ПЕРЕСТАВИТИ(X[j] , X[j+1]).
```

Сортування вставками. Нехай знову треба відсортувати елементи послідовності (масиву) $X_1, X_2,\ldots, X_n$ за зростанням. Суть методу вставок полягає в тому, що на і-му етапі ми «вставляємо» і-й елемент X_i у потрібну позицію серед елементів $X_1, X_2,\ldots,X_{i-1}$, які вже впорядковані. Пошук відповідної позиції здійснюється так, щоб послідовність елементів $X_1, X_2,\ldots,X_{i-1}, X_i$ стала впорядкованою. Доцільно ввести додатковий елемент X_0, значення ключа якого буде меншим за значення ключа будь-якого елемента $X_1, X_2,\ldots, X_n$. Часова складність методу вставок дорівнює $O(n^2)$ для послідовності з n елементів. Фрагмент цього алгоритму псевдомовою має вигляд

```
X[0]:= -∞;
for  i:=2 to n do
   begin
        j:=i:
        while  X[j-1]>X[j] do
        begin
             ПЕРЕСТАВИТИ(X[j-1] , X[j]);
             j:=j-1
        end
   end.
```

Сортування вибором. Ідея цього методу полягає в тому, що на і-му етапі сортування вибирається найменший елемент серед елементів $X_i,\ldots, X_n$, який переставляється місцями з елементом X_i. Після і-го етапу всі елементи $X_1,\ldots, X_i$ будуть упорядковані за зростанням. Часова складність методу вставок дорівнює $O(n^2)$ для послідовності з n елементів. Фрагмент цього алгоритму псевдомовою має вигляд

```
for  i:=1 to n-1 do
   begin
        index:=i;
        for  j:=i+1 to n do
             if  X[j]<X[index] then  index:=j;
        ПЕРЕСТАВИТИ(X[i], X[index])
   end.
```

Швидке сортування. Алгоритм швидкого сортування базується на методі «розділяй та володарюй» побудови алгоритмів. Основна ідея алгоритму полягає в тому, щоб на кожному кроці поділити елементи послідовності на дві частини, а потім кожну з частин сортувати окремо.

Розглянемо алгоритм швидкого сортування масиву X[1], X[2],…, X[n] за зростанням. З невеликими поправками його можна застосувати також до сортування файлу. Основними діями в алгоритмі швидкого сортування є:

1. вибір опорного елемента;
2. розбиття масиву або його частини на дві половини.

На початку роботи алгоритму в масиві вибирається деякий *опорний елемент* (ключ) V, відносно якого весь масив треба розділити на дві половини. Часто в якості опорного елемента V вибирається середній елемент масиву. Далі елементи масиву переставляються так, щоб для деякого індексу j усі переставлені елементи X[1],…, X[j] були меншими за V, а всі елементи X[j+1],…, X[n] були більшими або рівними V. Отримуємо дві половини X[1],…, X[j] та X[j+1],…, X[n] початкового масиву. Тепер застосовуємо описані дії до

кожної з частин X[1],…, X[j] та X[j+1],…, X[n] окремо і кожну з них теж розбиваємо на дві менші частини. Далі до отриманих менших частин знову застосовуємо описані дії і т. д. На кожному кроці роботи алгоритму розмір частин масиву зменшується. Врешті решт розмір усіх частин масиву не перевищуватиме 1, і алгоритм припинить роботу. Отримаємо упорядкований за зростанням масив. Алгоритм швидкого сортування є рекурсивним.

Нехай маємо якусь частину X[i],…, X[j] масиву. Для неї спочатку детально опишемо вибір опорного елемента V. У якості V виберемо більший із двох елементів масиву X[i],…, X[j], якщо він є. Пошук V починаємо з початку масиву. Функція, що повертає індекс опорного елемента V, має вигляд

```
function Find(i, j : integer) : integer;
var  k: integer;
begin
    for k:=i+1 to j do
            if X[k]>X[i] then
                        begin  Find:=k; exit  end
                else if X[k]<X[i] then
                                begin  Find:=i; exit  end;
    Find:=0;  { різних елементів немає }
end;
```

Тепер опишемо такі перестановки елементів масиву X[i],…, X[j], щоб зліва у масиві знаходилися елементи менші за опорний елемент V, а справа – більші або рівні. Нехай V=X[k], i ≤ k ≤j, причому індекс k буде знайдений за допомогою функції Find. Щоб виконати необхідні перестановки, введемо два вказівники L та R. Ці вказівники будуть пробігати по індексах масиву X[i],…, X[j]. Якщо L та R вибрані (зафіксовані), то вони будуть вказувати відповідно на лівий і правий кінці тієї частини масиву X, де в даний час ми переставляємо елементи. При цьому вважаємо, що вже всі елементи X[i],…, X[L-1], які розташовані зліва від X[L], мають значення менші за V. Відповідно елементи X[R+1],…, X[j], які розташовані справа від X[R], мають значення більші або рівні V. Нам необхідно переставляти елементи X[L],…, X[R].

Спочатку покладемо L=i та R=j. Далі будемо повторювати наступні дії, що переміщують вказівник L вправо, а вказівник R вліво до тих пір, поки вказівники не зустрінуться.

1. Вказівник (індекс елемента масиву) L зміщується вправо, поки не знайдеться елемент X[L], який не менший за опорний елемент V. Вказівник R зміщується вліво, поки не знайдеться елемент X[R], який менший за опорний елемент V.
2. Виконується перевірка: якщо L>R (на практиці можлива тільки ситуація, коли L=R+1), то перестановки елементів масиву X[i],…, X[j] закінчуються.
3. Якщо L<R (випадок L=R неможливий), то переставляємо місцями елементи X[L] та X[R]. Вказівник L зміщуємо вправо на одну позицію від попереднього положення, а вказівник R- на одну позицію вліво. Далі процес продовжується з пункту 1.

Наступна функція виконує описану перестановку елементів масиву X[i],…, X[j] та повертає індекс L, що вказує точку поділу цього масиву на дві частини відносно заданого опорного елемента V.

```
function  ПерестановкаМасиву(i, j: integer; V: < тип елементів масиву >) :
integer;
var  L, R : integer;
begin
   L:=i;    R:=j;
   repeat
         ПЕРЕСТАВИТИ(X[L], X[R]);
         while  X[L]<V  do L:=L+1;
         while  X[R]>=V  do R:=R-1;
```

```
            until  L>R;
            ПерестановкаМасиву:=L;
        end;
```

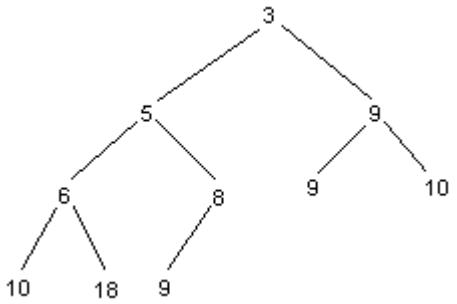
Тепер наведемо ескіз рекурсивної процедури швидкого сортування.

```
    procedure QuickSort( i, j : integer);
    var  V: < тип елементів масиву >;
            ІндексОпорногоЕлемента, k : integer;
    begin
     ІндексОпорногоЕлемента:= Find(i, j);
     if  ІндексОпорногоЕлемента <>0  then  {якщо всі елементи рівні, то нічого не робити}
            begin
                    V:=X[ІндексОпорногоЕлемента];
     `              k:= ПерестановкаМасиву(i, j, V);
                    QuickSort( i, k-1);
                    QuickSort( k, j);
            end;
    end;
```

Для сортування елементів усього масиву X[1], X[2],…, X[n] треба просто викликати процедуру QuickSort(1, n).

Часова складність алгоритму швидкого сортування для послідовності з n елементів дорівнює $O(n^2)$ в гіршому випадку. Але в середньому його ефективність дорівнює $O(n*\log_2 n)$. Тому з практичної точки зору він є дуже ефективним і найчастіше використовується при сортуванні.

Пірамідальне сортування. Пірамідою називається збалансоване за висотою бінарне дерево висоти h, у якому значення будь-якого вузла не більше (не менше) за значення його синів, і всі листки рівня h максимально зміщені вліво. Прикладом піраміди є зображене дерево. Висота цього дерева дорівнює 3, на третьому рівні розташовані листки 10, 18, 9, які максимально зміщені вліво, а попередні рівні дерева заповнені повністю. Таке дерево, тобто піраміду, можна повністю представити одновимірним масивом X[1], X[2],…, X[n]. У такому масиві елементи X[2*i], X[2*i+1] є відповідно лівим та правим синами елемента X[i]. Для зображеного дерева масив його вузлів буде такий



3, 5, 9, 6, 8, 9, 10, 10, 18, 9.

У піраміді найменший (найбільший) елемент розташований на вершині (у корені дерева), а у відповідному масиві він є першим. На зображеній піраміді значення кожного вузла-батька не більше за значення синів. Якщо будувати піраміду, користуючись відношенням «не менше», то на вершині піраміди буде розташований найбільший елемент.

Розглянемо суть алгоритму пірамідального сортування. Нехай задано масив X[1], X[2],…, X[n] і його треба відсортувати за спаданням. Для цього будемо використовувати піраміду з найменшим елементом на вершині, тобто піраміда буде будуватися з використанням відношення «не більше». Алгоритм пірамідального сортування складається з двох фаз: фаза побудови та фаза вибору.

Фаза побудови полягає в тому, щоб з масиву X[1], X[2],…, X[n] побудувати піраміду. Припустимо, що для цього масиву ми побудували збалансоване за висотою бінарне дерево, у якого всі листки найнижчого рівня максимально зміщені вліво (аналогічно зображеному дереву). Коренем цього дерева є X[1], вузлами першого рівня є

елементи X[2], X[3], вузлами другого рівня є елементи X[4], X[5], X[6], X[7] і т. д. Побудоване дерево не обов'язково є пірамідою, бо ми не знаємо, чи кожен вузол-батько є не більшим за своїх синів. Але друга половина масиву, а саме, X[n div 2 +1], X[n div 2 +2],…, X[n] у цьому дереві утворює піраміду, бо кожний із цих елементів не має синів. Використовуючи останній факт, перебудуємо дерево так, щоб отримати піраміду.

Отже, послідовність X[n div 2 +1], X[n div 2 +2],…, X[n] є пірамідою. Добавимо у її початок елемент X[n div 2] і отримаємо послідовність

X[n div 2], X[n div 2 +1], X[n div 2 +2],…, X[n],

яку перебудуємо у піраміду. Потім добавляємо наступний попередній елемент і послідовність

X[n div 2-1], X[n div 2], X[n div 2 +1], X[n div 2 +2],…, X[n]

знову перебудуємо у піраміду. Продовжуючи цей процес далі, із масиву X[1], X[2],…, X[n] побудуємо піраміду. Таким чином, на кожному кроці фази побудови піраміди до послідовності X[i+1],…, X[n], яка є пірамідою, добавляємо попередній елемент масиву X[i] та утворюємо послідовність X[i], X[i+1],…, X[n], яку перебудовуємо у піраміду.

Опишемо цей крок перебудови детальніше. Після добавлення елемента X[i] до піраміди X[i+1],…, X[n] утворюється послідовність X[i], X[i+1],…, X[n], яка вже не обов'язково є пірамідою. Порушив піраміду елемент X[i], тому його треба «проштовхнути» вниз по дереву, щоб утворилася піраміда. Для цього порівнюємо X[i] з меншим із його синів X[2*i], X[2*i+1]. Якщо X[i] не більший за синів, то нічого робити не треба, бо послідовність X[i], X[i+1],…, X[n] вже є пірамідою. Якщо X[i] більший за якогось із синів, то переставляємо X[i] із меншим із синів. Елемент X[i] опинився на новому місці, і для нього можуть існувати наступні сини. Якщо X[i] знову порушує піраміду, то його ще раз треба переставити з меншим із синів. Елемент X[i] потрібно переставляти («проштовхувати») до тих пір, поки не отримаємо піраміду. Процедура проштовхування має вигляд:

```
procedure ПРОШТОВХНУТИ( i, j: integer);
      {Елементи X[i+1],…, X[j] утворюють піраміду, а елемент X[i] добавляється.
       Процедура перебудовує послідовність X[i], X[i+1],…,X[j] у піраміду}
var r: integer;          {вказує поточну позицію елемента X[i]}
begin
 r:=i;     {початкова ініціалізація}
 while r <= j div 2  do
        if j = 2*r then
        begin  {елемент у позиції r має одного сина в позиції 2*r}
               if X[r]>X[2*r] then ПЕРЕСТАВИТИ(X[r], X[2*r]);
               r:=j   {передчасний вихід із циклу while}
        end
        else {елемент у позиції r має двох синів у позиціях 2*r та 2*r+1}
               if X[r]>X[2*r] and X[2*r]<=X[2*r+1] then
                      begin  {перестановка елемента в позиції r з лівим сином}
                             ПЕРЕСТАВИТИ(X[r], X[2*r]);
                             r:= 2*r
                      end
               else if X[r]>X[2*r+1] and X[2*r+1]<X[2*r]  then
                      begin  {перестановка елемента в позиції r з правим сином}
                             ПЕРЕСТАВИТИ(X[r], X[2*r+1]);
                             r:= 2*r+1
                      end
               else {елемент в позиції r не порушує піраміду}
                      r:= j   {вихід із циклу while}
end;
```

Із використанням цієї процедури фаза побудови піраміди з масиву X[1], X[2],…, X[n] має вигляд **for** i:= n div 2 **downto** 1 **do** ПРОШТОВХНУТИ(i, n).

Розглянемо тепер другу фазу алгоритму – фазу вибору. У результаті виконання першої фази отримали піраміду X[1], X[2],…, X[n]. Переставляємо елементи X[1], X[n] місцями. Таким чином, на останньому місці буде знаходитися найменший елемент. Тепер розглядаємо послідовність X[1], X[2],…, X[n-1] без останнього елемента. Якщо вона не є пірамідою, то порушує піраміду елемент X[1]. Застосувавши до послідовності X[1], X[2],…, X[n-1] процедуру ПРОШТОВХНУТИ, перетворюємо цю послідовність у піраміду. У піраміді X[1], X[2],…, X[n-1] знову переставляємо елементи X[1], X[n-1] місцями. Продовжуючи цей процес далі та зменшуючи на кожному кроці послідовність, яку треба перебудовувати у піраміду, отримаємо відсортований початковий масив за спаданням. Весь алгоритм пірамідального сортування масиву X[1], X[2],…, X[n] має вигляд:

```
    {Фаза побудови}
for  i:= n div 2  downto  1 do ПРОШТОВХНУТИ(i, n).
    {Фаза вибору}
for  i:= n  downto  2 do
begin
    ПЕРЕСТАВИТИ(X[1], X[i]);
    ПРОШТОВХНУТИ(i, n).
end
```

Часова складність алгоритму пірамідального сортування для послідовності з n елементів дорівнює $O(n*\log_2 n)$ як у гіршому випадку так і в середньому. Хоч він є дуже ефективним, але на практиці віддають перевагу швидкому сортуванню.

Сортування методом злиття. Нехай дано два упорядковані масиви (файли) $X[1] \le X[2] \le \ldots \le X[n]$ та $Y[1] \le Y[2] \le \ldots \le Y[m]$. В основі сортування методом злиття лежить об'єднання цих масивів у один відсортований масив $Z[1] \le Z[2] \le \ldots \le Z[n+m]$. Для цього масиви X та Y паралельно переглядаються, і на кожному кроці менше з двох значень одного з масивів заноситься до Z. Фрагмент цього алгоритму має вигляд:

```
i:=1;  j:=1;
while  (i<=n) and (j<=m) do
    if X[i]<Y[j]  then
        begin  Z[i+j-1]:=X[i];  i:=i+1  end
            else
        begin  Z[i+j-1]:=Y[j];  j:=j+1  end;
    for i:=i to n do  Z[j+i-1]:=X[i];          {якщо масив X не вичерпано}
    for j:=j to m do Z[i+j-1]:=Y[j];           {якщо масив Y не вичерпано}
```

Дж. фон Нейман запропонував метод сортування злиттям, у якому реалізовано принцип «розділяй та володарюй». Масив ділиться навпіл, до кожної половини застосовується рекурсивно та сама процедура сортування злиттям, а відсортовані частини з'єднуються в один упорядкований масив. Отже, базовою операцією методу є злиття двох упорядкованих масивів у один. Час злиття упорядкованих масивів лінійно залежить від їх сумарної довжини. У цілому, алгоритм фон Неймана потребує виконання $O(n*\log_2 n)$ базових операцій над елементами масиву розмірності n.

Приклад 1

Записати у файл інформацію про декількох студентів: прізвище, ім'я та рік народження. Скласти програму для впорядкування даних кожного з полів, використовуючи різні методи сортування.

Аналіз задачі

Організуємо створення двійкового файлу структур у вигляді функції. Потім обчислимо кількість структур у файлі, бо ця величина потрібна для сортування файлу. Для сортування файлу за прізвищем використаємо метод вибору, за іменем – метод бульбашок, за роком народження – метод вставок.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
// Визначення типу структури
typedef struct inf
        {       char prizv[20];             // прізвище
                char imja[10];              // ім'я
                int rik;                    // рік народження
        }       anketa;
// Функція для заповнення полів структури anketa. Через вказівник p із функції у програму
// передається заповнена структура.
void StvorenStruct(anketa *p);
// Функція створює файл структур. Через вказівник p передається файл.
void StvorenFile(FILE *p);
// Функція виводить на екран вміст файлу.
void DrucFile(FILE *p);
// Функція повертає кількість структур у файлі, що пов'язаний з вказівником p.
int FileSize(FILE *p);
// Функція виконує сортування файлу структур методом вибору за прізвищами студентів в
// алфавітному порядку. Через параметр k передається кількість структур.
void SortFileVybor(FILE *p, int k);
// Функція виконує сортування файлу структур методом бульбашок за іменами студентів в
// алфавітному порядку. Через параметр k передається кількість структур.
void SortFileBuble(FILE *p, int k);
// Функція виконує сортування файлу структур методом вставок за зростанням років
// народження студентів. Через параметр k передається кількість структур.
void SortFileVstavka(FILE *p, int k);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int l;                              // Змінна для кількості структур у файлі
        char filename[120];                 // Змінна для імені файлу
        printf ("Введіть імя файлу\n");
        gets(filename);                     // Введення імені файлк з клавіатури
        FILE * fp;                          // Вказівник на потік (файл)
        fp=fopen(filename, "wb");           // Відкриття двійкового файлу для створення
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        StvorenFile(fp);                    // Створення файлу
        fp=fopen(filename, "r+b");          // Відкриття існуючого двійкового файлу для
                                                читання і запису
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
```

```
	}
	printf ("Друк файлу\n");
	DrucFile(fp);			// Виведення на екран вмісту файлу
	fseek(fp, 0, SEEK_SET);		// Встановлення поточного вказівника на початок файлу
	l=FileSize(fp);			// Обчислення кількості структур у файлі
	SortFileVybor(fp, l);		// Сортування файлу за прізвищем
	fseek(fp, 0, SEEK_SET);		// Встановлення поточного вказівника на початок файлу
	printf ("Друк відсортованого файлу методом вибору (за прізвищем)\n");
	DrucFile(fp);			// Виведення відсортованого файлу
	SortFileBuble(fp, l);		// Сортування файлу за іменем
	fseek(fp, 0, SEEK_SET);		// Встановлення поточного вказівника на початок файлу
	printf ("Друк відсортованого файлу методом бульбашок (за іменем)\n");
	DrucFile(fp);			// Виведення відсортованого файлу
	SortFileVstavka(fp, l);		// Сортування файлу за роком народження
	fseek(fp, 0, SEEK_SET);		// Встановлення поточного вказівника на початок файлу
	printf ("Друк відсортованого файлу методом вставок (за роком народження)\n");
	DrucFile(fp);			// Виведення відсортованого файлу
	fclose(fp);			// Закриття файлу
	system("pause");
	return 0;
}
//-----------------------------------------------------------------------------------------------------------
void StvorenStruct(anketa *p)
{	// Заповнення з клавіатури полів структури anketa
	fflush(stdin);			// Очищення буфера стандартного потоку введення
	printf ("Введіть прізвище:  ");
	gets(p->prizv);			// Введення прізвища
	printf ("Введіть імя:  ");
	gets(p->imja);			// Введення імені
	printf ("Введіть рік народження:  ");
	scanf("%d", &(p->rik));		// Введення року народження
}
//-----------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{	// Запис у файл структур. Цикл припиняє роботу, якщо вводиться структура, у якій
	// прізвище є рядком "#". Ця структура не записується у файл.
	printf ("Введіть записи у файл. В останньому записі прізвище - #\n");
	anketa x;			// Структурна змінна
	StvorenStruct(&x);		// Введення інформації про студента
	while (strcmp(x.prizv, "#")!=0)
	{	fwrite(&x, sizeof(anketa), 1, p);	// Запис цієї інформації у файл
		StvorenStruct(&x);
	}
	fclose(p);				// Закриття файлу
}
//-----------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{	anketa x;
	fread(&x, sizeof(anketa), 1, p);	// Читаємо з файлу анкету студента
	while (!feof(p))			// Поки не кінець файлу
	{	printf ("%25s%15s%10d\n", x.prizv, x.imja, x.rik);		// Виведення анкети
		fread(&x, sizeof(anketa), 1, p);	// Знову читаємо з файлу анкету студента
```

```
        }
}
//-------------------------------------------------------------------------------------------------------------
int FileSize(FILE *p)
{       // Підрахунок кількості структур у файлі
        int l=0;                                // Початкове значення лічильника
        anketa x;
        fread(&x, sizeof(anketa), 1, p);        // Читаємо з файлу анкету студента
        while (!feof(p))                        // Поки не кінець файлу
        {       l++;                            // Збільшуємо лічильник
                fread(&x, sizeof(anketa), 1, p);        // Знову читаємо з файлу анкету студента
        }
        return l;                               // Повертаємо значення лічильника
}
//-------------------------------------------------------------------------------------------------------------
void SortFileVybor(FILE *p, int k)
{       // Сортування файлу методом вибору за прізвищем. У відсортованому файлі
        // прізвища розташовуються в алфавітному порядку.
        int i, j, index;
        anketa x, y;
        char pr[20];
        for (i=0; i<k-1; i++)
        {       index=i;
                fseek(p, i*sizeof(anketa), SEEK_SET);       // Поточний вказівник файлу
                                                                встановлюємо на і-ту анкету
                fread(&x, sizeof(anketa), 1, p);                // Читаємо цю анкету з файлу
                strcpy(pr, x.prizv);                // Копіюємо прізвище з цієї анкети в рядок pr
                for (j=i+1; j<k; j++)
                {       fread(&y, sizeof(anketa), 1, p);        // Читаємо j-ту анкету з файлу
                        if (strcmp(pr, y.prizv)>0)              // Порівнюємо прізвища
                        {       strcpy(pr, y.prizv);    // Менше прізвище копіюємо в рядок pr
                                index=j;  // Запам'ятовуємо номер анкети з меншим прізвищем
                        }
                }
                fseek(p, index*sizeof(anketa), SEEK_SET);           // Поточний вказівник файлу
                                        встановлюємо на анкету з найменшим прізвищем
                fread(&y, sizeof(anketa), 1, p);                // Читаємо цю анкету з файлу
                // Перестановка анкет
                fseek(p, i*sizeof(anketa), SEEK_SET);
                fwrite(&y, sizeof(anketa), 1, p);
                fseek(p, index*sizeof(anketa), SEEK_SET);
                fwrite(&x, sizeof(anketa), 1, p);
        }
}
//-------------------------------------------------------------------------------------------------------------
void SortFileBuble(FILE *p, int k)
{       // Сортування файлу методом бульбашок за іменем. У відсортованому файлі
        // імена розташовуються в алфавітному порядку
        int i, j;
        anketa x, y;
        for (i=k-2; i>=0; i--)
                for (j=0; j<=i; j++)
```

```
                {       fseek(p, j*sizeof(anketa), SEEK_SET);       // Поточний вказівник файлу
                                                                    встановлюємо j-ту анкету
                        fread(&x, sizeof(anketa), 1, p);            // Читаємо j-ту анкету
                        fread(&y, sizeof(anketa), 1, p);            // Читаємо (j+1)-ту анкету
                        if (strcmp(x.imja, y.imja)>0)               // Порівнюємо імена
                        {       // Перестановка анкет
                                fseek(p, j*sizeof(anketa), SEEK_SET);
                                fwrite(&y, sizeof(anketa), 1, p);
                                fwrite(&x, sizeof(anketa), 1, p);
                        }
                }
}
//-------------------------------------------------------------------------------------------------------------
void SortFileVstavka(FILE *p, int k)
{       // Сортування файлу методом вставок за зростанням року народження.
        anketa x, y;
        for (i=1; i<k; i++)
        {       j=i;
                fseek(p, (j-1)*sizeof(anketa), SEEK_SET);  // Поточний вказівник файлу
                                                           встановлюємо (j-1)-у анкету
                fread(&x, sizeof(anketa), 1, p);           // Читаємо (j-1)-ту анкету
                fread(&y, sizeof(anketa), 1, p);           // Читаємо j-ту анкету
                while ((x.rik > y.rik) && (j>0))
                {       // Перестановка анкет
                        fseek(p, (j-1)*sizeof(anketa), SEEK_SET);
                        fwrite(&y, sizeof(anketa), 1, p);
                        fwrite(&x, sizeof(anketa), 1, p);
                        j--;
                        if (j>0)
                        {       // Читаємо наступну пару анкет
                                fseek(p, (j-1)*sizeof(anketa), SEEK_SET);
                                fread(&x, sizeof(anketa), 1, p);
                                fread(&y, sizeof(anketa), 1, p);
                        }
                }
        }
}
```

```
Введіть імя файлу
D:\fff
Введіть записи у файл. В останньому записі прізвище - #
Введіть прізвище:  Ivanov
Введіть імя:  Ivan
Введіть рік народження:  1990
Введіть прізвище:  Rak
Введіть імя:  Petro
Введіть рік народження:  1989
Введіть прізвище:  Bober
Введіть імя:  Oksana
Введіть рік народження:  1993
Введіть прізвище:  Petrov
Введіть імя:  Danylo
Введіть рік народження:  1992
Введіть прізвище:  Strixa
Введіть імя:  Petro
Введіть рік народження:  1987
Введіть прізвище:  Soroka
Введіть імя:  Marija
Введіть рік народження:  1990
Введіть прізвище:  #
Введіть імя:  #
Введіть рік народження:  1
```

Результат роботи програми.

Приклад 2

Створити файл дійсних чисел та упорядкувати його за зростанням методом швидкого сортування.

Аналіз задачі

Файл дійсних чисел створимо двійковим та застосуємо до нього алгоритм швидкого сортування.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
// Функція створює двійковий файл дійсних чисел, який передається через вказівник p.
void StvorenFile(FILE *p);
// Функція виводить вміст двійкового файлу дійсних чисел на екран.
void DrucFile(FILE *p);
// Функція обчислює кількість дійсних чисел у файлі та повертає цей результат.
int FileSize(FILE *p);
// Реалізація функції function Find(i, j : integer) : integer, що описана в теоретичній частині.
// Дана функція знаходить номер опорного елемента (дійсного числа) та повертає його.
// Опорний елемент визначається із послідовності дійсних чисел двійкового файлу з
// номерами від і до j. Якщо всі числа цієї послідовності однакові, то функція повертає -1,
// бо нумерація елементів двійкового файлу починається з нуля.
int Find(FILE *p, int i, int j);
// Реалізація функції
//         function  ПерестановкаМасиву(i, j: integer; V: < тип елементів масиву >) : integer,
// що описана в теоретичній частині. Дана функція переставляє дійсні числа двійкового
// файлу з номерами від і до j так, що вони утворять дві частини відносно опорного числа.
// У лівій частині знаходяться числа, які менші опорного числа, а в правій – інші. Через
// параметр k у функцію передається номер опорного елемента. Функція повертає межу
// (номер) розбиття заданої послідовності чисел.
int PerestanovkaMasivu(FILE *p, int i, int j, int k);
```

// Функція реалізує алгоритм швидкого сортування для послідовності дійсних чисел двійкового файлу з номерами від i до j.
void QuikSort(FILE *p, int i, int j);

```
int _tmain(int argc, _TCHAR* argv[])
{	setlocale (LC_ALL,"RUS");
	int n;				// Кількість дійсних чисел у двійковому файлі
	int nv;				// Номер опорного елемента
	char filename[120];		// Змінна для імені файлу
	printf ("Введіть імя файлу\n");
	gets(filename);			// Введення імені файлу з клавіатури
	FILE * fp;
	fp=fopen(filename, "wb");	// Відкриття двійкового файлу для запису
	if (fp==NULL)
	{	puts("Файл не відкритий\n");
		exit(0);
	}
	StvorenFile(fp);		// Створення двійкового файлу дійсних чисел
	fp=fopen(filename, "r+b");	// Відкриття двійкового файлу для читання і запису
	if (fp==NULL)
	{	puts("Файл не відкритий\n");
		exit(0);
	}
	printf ("Друк файлу\n");
	DrucFile(fp);			// Виведення вмісту файлу на екран
	fseek(fp, 0, SEEK_SET);		// Встановлення поточного вказівника файлу на початок
	n=FileSize(fp);			// Визначення кількості дійсних чисел у файлі
	QuikSort(fp, 0, n-1);		// Сортування файлу за зростанням
	fseek(fp, 0, SEEK_SET);		// Встановлення поточного вказівника файлу на початок
	printf ("Друк відсортованого за зростанням файлу\n");
	DrucFile(fp);			// Виведення вмісту відсортованого файлу на екран
	fclose(fp);			// Закриття файлу
	system("pause");
	return 0;
}
//-----------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{	// Функція створює двійковий файл дійсних чисел. Числа вводяться з клавіатури в
	// циклі, який припиняє роботу, коли введене число 999
	printf ("Введіть дійсні числа у файл. Останне число - 999\n");
	double x;			// Змінна для дійсного числа
	scanf("%lf", &x);		// Вводимо число з клавіатури
	while (x!=999)
	{	fwrite(&x, sizeof(double), 1, p);	// Записуємо число у файл
		scanf("%lf", &x);	// Знову вводимо число з клавіатури
	}
	fclose(p);			// Закриття файлу
}
//-----------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{	// Функція виводить на екран вміст двійкового файлу дійсних чисел. Читання файлу
	// здійснюється в циклі, який припиняє роботу, коли досягнутий кінець файлу
```

```
        double x;                                   // Змінна для дійсного числа
        fread(&x, sizeof(double), 1, p);            // Читаємо з файлу число
        while (!feof(p))                            // Поки не кінець файлу
        {       printf("%10.3lf", x);               // Виводимо число на екран
                fread(&x, sizeof(double), 1, p);    // Знову читаємо з файлу число
        }
        printf("\n");
}
//-------------------------------------------------------------------------------------------------------------
int FileSize(FILE *p)
{       // Функція обчислює кількість дійсних у файлі. Читання файлу здійснюється в
        // циклі, який припиняє роботу, коли досягнутий кінець файлу
        int l=0;                                    // Лічильник дійсних чисел
        double x;                                   // Змінна для дійсного числа
        fread(&x, sizeof(double), 1, p);            // Читаємо з файлу число
        while (!feof(p))                            // Поки не кінець файлу
        {       l++;                                // Збільшуємо лічильник
                fread(&x, sizeof(double), 1, p);    // Знову читаємо з файлу число
        }
        return l;                                   // Повертаємо значення лічильника
}
//-------------------------------------------------------------------------------------------------------------
int Find(FILE *p, int i, int j)
{       // Функція знаходить номер опорного елемента в послідовності чисел файлу з
        // номерами від i до j
        int k;
        double x, y;                                // Змінні для дійсних чисел
        fseek(p, i*sizeof(double), SEEK_SET);       // Встановлюємо поточний вказівник
                                                        файлу на число з номером i
        fread(&x, sizeof(double), 1, p);            // Читаємо це число з файлу та
                                                        запам’ятовуємо його в змінній x
        for(k=i+1; k<=j; k++)
        {       fread(&y, sizeof(double), 1, p);    // Читаємо наступне число з файлу та
                                                        запам’ятовуємо його в змінній y
                if (y>x)
                        return k;                   // Знайдений номер опорного елемента
                else
                        if (y<x)
                                return i;           // Знайдений номер опорного елемента
        }
        return -1;                                  // Опорного елемента немає
}
//-------------------------------------------------------------------------------------------------------------
int PerestanovkaMasivu(FILE *p, int i, int j, int k)
{       // Функція розбиває послідовність чисел файлу з номерами від i до j на дві частини
        // по відношенню до опорного елемента. У лівій частині будуть знаходитися числа,
        // що менші опорного числа, а в правій – більші або рівні. Параметр i задає ліву
        // межу послідовності, j – праву межу, k – номер опорного елемента.
        int l, r;               // Лівий та правий вказівники на номери чисел
        double v;               // Змінна для опорного елемента
        double x, y;                                // Змінні для дійсних чисел
```

```
        fseek(p, k*sizeof(double), SEEK_SET);      // Встановлюємо поточний вказівник
                                                        файлу на опорний елемент
        fread(&v, sizeof(double), 1, p);           // Читаємо цей елемент з файлу та
                                                        запам'ятовуємо його в змінній v
        l=i;                                       // Початкове значення лівого вказівника
        r=j;                                       // Початкове значення правого вказівника
        do
        {       fseek(p, l*sizeof(double), SEEK_SET);      // Встановлюємо поточний
                                                        вказівник файлу на число з номером l
                fread(&x, sizeof(double), 1, p);           // Читаємо це число
                fseek(p, r*sizeof(double), SEEK_SET);      // Встановлюємо поточний
                                                        вказівник файлу на число з номером r
                fread(&y, sizeof(double), 1, p);           // Читаємо це число
                // Перестановка чисел з номерами l та r
                fseek(p, l*sizeof(double), SEEK_SET);
                fwrite(&y, sizeof(double), 1, p);
                fseek(p, r*sizeof(double), SEEK_SET);
                fwrite(&x, sizeof(double), 1, p);
                // Рух вказівника l зліва направо для пошуку числа, яке більше або рівне за
                // опорний елемент
                fseek(p, l*sizeof(double), SEEK_SET);      // Встановлюємо поточний
                                                        вказівник файлу на число з номером l
                fread(&x, sizeof(double), 1, p);       // Читаємо це число
                while (x<v)                            // Пошук числа більшого або рівного за v
                {       l++;                           // Збільшуємо вказівник
                        fread(&x, sizeof(double), 1, p);       // Читаємо наступне число
                }
                // Рух вказівника  r справа на ліво для пошуку числа, яке менше за
                // опорний елемент
                fseek(p, r*sizeof(double), SEEK_SET);      // Встановлюємо поточний
                                                        вказівник файлу на число з номером r
                fread(&y, sizeof(double), 1, p);       // Читаємо це число
                while (y>=v)                           // Пошук числа меншого за v
                {       r--;                           // Зменшуємо вказівник
                        fseek(p, r*sizeof(double), SEEK_SET);      // Встановлюємо поточний
                                                        вказівник файлу на число з номером r
                        fread(&y, sizeof(double), 1, p);       // Читаємо наступне число
                }
        }
        while (l<=r);
        return l;               // Повертаємо номер елемента, який визначає точку розбиття
}
//-------------------------------------------------------------------------------------------------------------------
void QuikSort(FILE *p, int i, int j)
{       // Функція реалізує алгоритм швидкого сортування для послідовність чисел файлу з
        // номерами від i до j
        int IndexOpornogoElementa;
        int k;
        IndexOpornogoElementa=Find(p, i, j);       // Визначення номера опорного числа
        if (IndexOpornogoElementa>=0)
        {       k=PerestanovkaMasivu(p, i, j, IndexOpornogoElementa);  // Розбиття
                                                                послідовності на дві части
```

```
            QuikSort(p, i, k-1);    // Застосовуємо швидке сортування до лівої частини
            QuikSort(p, k, j);      // Застосовуємо швидке сортування до правої частини
        }
}
```

```
Введіть імя файлу
D:\fff
Введіть дійсні числа у файл. Останне число - 999
5,6 76,99 -23,87 4,5 12,78 6 -8
89,6 -11,11 0,56 -0,56 78
999
Друк файлу
     5,600    76,990   -23,870     4,500    12,780     6,000    -8,000    89,600
   -11,110     0,560    -0,560    78,000
Друк відсортованого за зростанням файлу
   -23,870   -11,110    -8,000    -0,560     0,560     4,500     5,600     6,000
    12,780    76,990    78,000    89,600
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 3

Створити файл структур із полями: прізвище, ім’я, по батькові та рік народження. Упорядкувати файл за алфавітом прізвищ, імен, по-батькові.

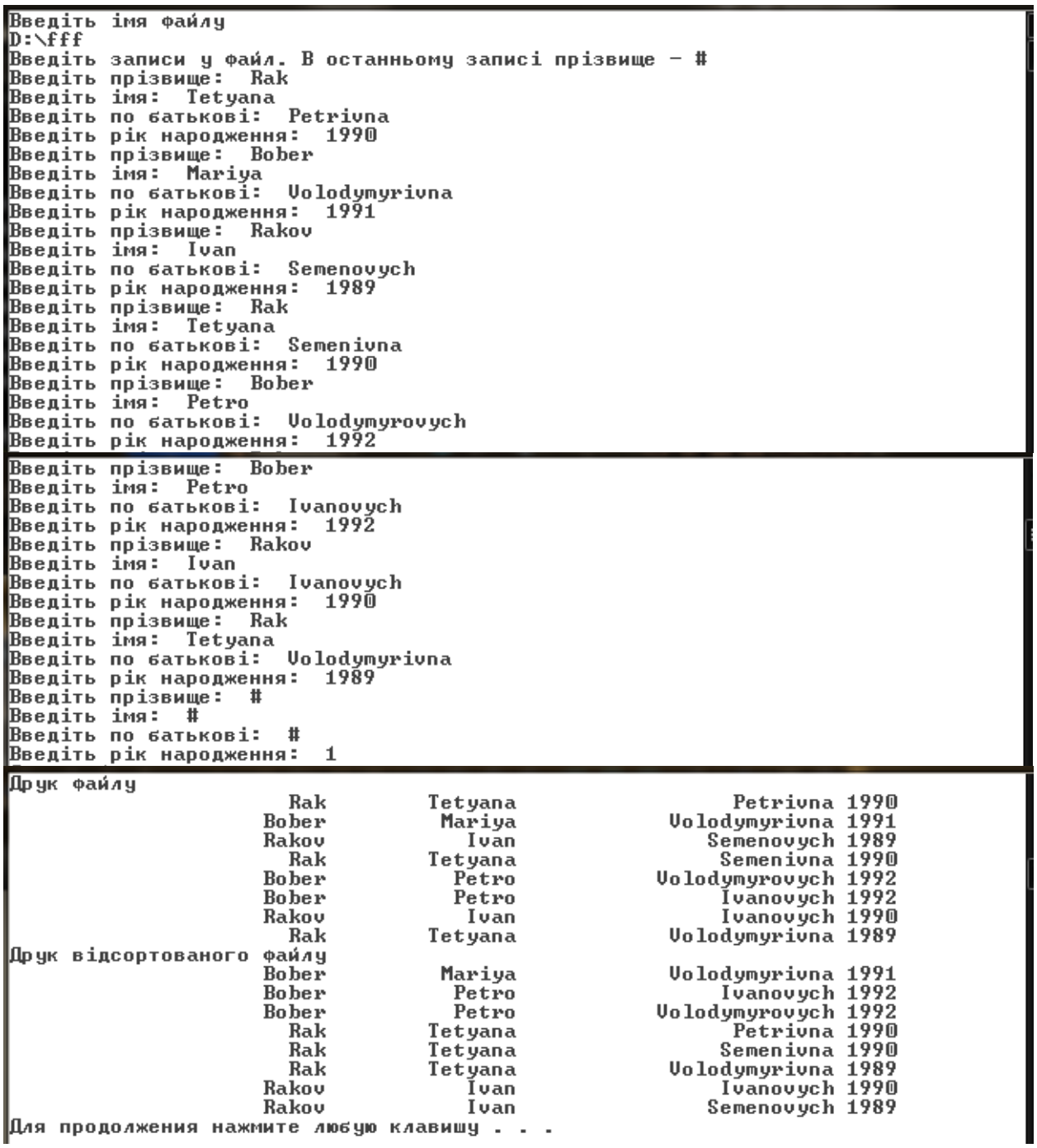
Аналіз задачі

Створення двійкового файлу структур організуємо у вигляді функції. У файл будемо заносити структури до тих пір, поки не зустрінеться запис, у якому прізвище рівне ’#’. Також створимо функцію для виведення структур файлу на екран. Основною буде функція сортування файл. Для сортування використаємо алгоритм бульбашок, оскільки його код невеликий. Упорядковуватимемо файл за алфавітом прізвищ. При однакових прізвищах проводитимемо сортування за іменами, а якщо співпадають імена, то сортуватимемо за по-батькові.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
// Визначення типу anketa
typedef struct inf
        {       char prizv[20];                 // прізвище
                char imja[10];                  // ім’я
                char batko[20];                 // по-батькові
                int rik;                        // рік народження
        }       anketa;
// Функція для заповнення полів структури типу anketa. Через вказівник p у програму
// передається заповнена анкета
void StvorenStruct(anketa *p);
// Функція створює файл структур.
void StvorenFile(FILE *p);
// Функція виводить на екран вміст файлу структур.
void DrucFile(FILE *p);
// Функція обчислює і повертає кількість структур у файлі.
int FileSize(FILE *p);
// Функція виконує сортування файлу.
void SortFile(FILE *p, int k);

int _tmain(int argc, _TCHAR* argv[])
```

```
{       setlocale (LC_ALL,"RUS");
        int l;                                  // Змінна для кількості структур у файлі
        char filename[120];                     // Змінна для імені файлу
        printf ("Введіть імя файлу\n");
        gets(filename);                         // Введення імені файлу з клавіатури
        FILE * fp;                              // Оголошення вказівника на потік
        fp=fopen(filename, "wb");               // Відкриття двійкового файлу для запису
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        StvorenFile(fp);                        // Створення двійкового файлу стуктур
        fp=fopen(filename, "r+b");    // Відкриття двійкового файлу для запису і читання
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        printf ("Друк файлу\n");
        DrucFile(fp);                   // Виведення вмісту двійкового файлу структур на екран
        fseek(fp, 0, SEEK_SET);         // Встановлення поточного вказівника на початок файлу
        l=FileSize(fp);                 // Обчислення кількості структур у файлі
        SortFile(fp, l);                // Сортування файлу
        fseek(fp, 0, SEEK_SET);         // Встановлення поточного вказівника на початок файлу
        printf ("Друк відсортованого файлу\n");
        DrucFile(fp);                   // Виведення відсортованого файлу
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------
void StvorenStruct(anketa *p)
{       // Заповнення полів структури з клавіатури
        fflush(stdin);                          // Очищення буфера потоку введення
        printf ("Введіть прізвище:  ");
        gets(p->prizv);                         // Введення прізвища
        printf ("Введіть імя:  ");
        gets(p->imja);                          // Введення імені
        printf ("Введіть по батькові:  ");
        gets(p->batko);                         // Введення по-батькові
        printf ("Введіть рік народження:  ");
        scanf("%d", &(p->rik));                 // Введення року народження
}
//-------------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{       // Створення двійкового файлу структур. Введення структур організовано в циклі
        // до тих пір, поки не буде введена структура з прізвищем "#". Вона не записується
        // у файл.
        printf ("Введіть записи у файл. В останньому записі прізвище - #\n");
        anketa x;
        StvorenStruct(&x);                              // Заповнення структури
        while (strcmp(x.prizv, "#")!=0)
        {       fwrite(&x, sizeof(anketa), 1, p);       // Запис її у файл
                StvorenStruct(&x);                      // Заповнення наступної структури
```

```
        }
        fclose(p);                                  // Закриття файлу
}
//-----------------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{       // Виведення файлу на екран.
        anketa x;
        fread(&x, sizeof(anketa), 1, p);            // Читаємо структуру з файлу
        while (!feof(p))                            // Поки не кінець файлу
        {       // Виводимо структуру на екран
                printf ("%25s%15s%25s%5d\n", x.prizv, x.imja, x.batko, x.rik);
                fread(&x, sizeof(anketa), 1, p);    // Читаємо наступну структуру з файлу
        }
}
//-----------------------------------------------------------------------------------------------------------------
int FileSize(FILE *p)
{       // Підрахунок кількості структур у файлі
        int l=0;                                    // Лічильник структур
        anketa x;
        fread(&x, sizeof(anketa), 1, p);            // Читаємо структуру з файлу
        while (!feof(p))                            // Поки не кінець файлу
        {       l++;                                // Збільшуємо лічильник
                fread(&x, sizeof(anketa), 1, p);    // Читаємо наступну структуру з файлу
        }
        return l;                                   // Повертаємо значення лічильника
}
//-----------------------------------------------------------------------------------------------------------------
void SortFile(FILE *p, int k)
{       // Сортування двійкового файлу структур за алфавітом прізвищ, імен, по батькові.
        // Для сортування використовується метод бульбашок. Через параметр k у функцію
        //передається кількість структур у файлі.
        int i, j;
        anketa x, y;                                // Змінні для структур
        for (i=k-2; i>=0; i--)
                for (j=0; j<=i; j++)
                {       fseek(p, j*sizeof(anketa), SEEK_SET);       // Встановлення поточного
                                                                    вказівника на структуру з номером j
                        fread(&x, sizeof(anketa), 1, p);        // Читаємо j-ту структуру
                        fread(&y, sizeof(anketa), 1, p);        // Читаємо (j+1)-у структуру
                        // Якщо j-те прізвище більше (j+1)-го або прізвища рівні та j-те ім’я
                        // більше (j+1)-го або прізвища й імена рівні та j-те по батькові більше
                        // (j+1)-го
                        if ((strcmp(x.prizv, y.prizv)>0)||
                        ((strcmp(x.prizv, y.prizv)==0) && (strcmp(x.imja, y.imja)>0))||
((strcmp(x.prizv, y.prizv)==0)&&(strcmp(x.imja, y.imja)==0)&&(strcmp(x.batko, y.batko)>0)))
                        {       // Виконуємо перестановку структур j-ї та (j+1)-ї
                                fseek(p, j*sizeof(anketa), SEEK_SET);
                                fwrite(&y, sizeof(anketa), 1, p);
                                fwrite(&x, sizeof(anketa), 1, p);
                        }
                }
}
```



Результат роботи програми.

Задачі

1. Створити файл дійсних чисел та упорядкувати його за спаданням методом пірамідального сортування.
2. Створити файл структур із полями: прізвище, ім’я, по-батькові та рік народження. Упорядкувати файл за алфавітом прізвищ, імен, по батькові методом швидкого сортування.

☺ Індивідуальні завдання

Основний рівень

1. Перетворити масив X за наступним правилом : усі від'ємні елементи масиву X перенести на початок, а всі інші - у кінець. Усі від'ємні елементи відсортувати за не спаданням.
2. Дано дійсні числа $a_1, a_2, ..., a_n$. Серед цих чисел усі від'ємні числа збільшити на 0,25, а всі невід'ємні замінити на 0,2. Потім отримані від'ємні числа відсортувати за не зростанням.
3. Дано два масиви чисел A та B по N елементів у кожному. Упорядкувати масив Z за спаданням елементів, за умови, що елементи масиву Z утворюються за формулою $z_i = a_i - b_i \sin(i)$.
4. Обчислити координати вектора $Z = \{z_1, z_2, ..., z_N\}$ і впорядкувати їх за зростанням, якщо $z_k = x_k + m y_k$, де x_k, y_k - координати заданих векторів X та Y, а m визначається умовою:
$$m = \begin{cases} k & \text{при } |\sin k| \le 0.3 \\ \sqrt{k} & \text{при } |\sin k| > 0.3 \end{cases}$$
5. Дано двовимірний масив розмірності NxM. Елементами масиву є цілі числа. Упорядкувати рядки за не спаданням елементів першого стовпця.
6. Дано двовимірний масив розмірності NxM. Елементами масиву є дійсні числа. Упорядкувати рядки за не зростанням елементів другого стовпчика.
7. Дано двовимірний масив розмірності NxM. Елементами масиву є цілі числа. Упорядкувати стовпці за не зростанням елементів першого рядка.
8. Дано двовимірний масив розмірності NxM. Елементами масиву є дійсні числа. Упорядкувати рядки за не спаданням сум елементів рядків.
9. Дано цілі числа $x_1, ..., x_n$. Отримати в порядку зростання всі різні числа, що входять до $x_1, ..., x_n$. При сортуванні необхідно відкинути елементи, що вже зустрічалися раніше. Якщо при пошуку місця для x_i в упорядкованій послідовності $x_1, ..., x_к$ (к<i) виявиться, що серед $x_1, ..., x_к$ є елемент, рівний x_i, то слід перейти до розгляду x_{i+1} без зміни $x_1, ..., x_к$.
10. Дано дійсні числа $c_1, ..., c_n$, $d_1, ..., d_k$ ($c_1 \le c_2 \le ... \le c_n$, $d_1 \le d_2 \le ... \le d_k$). Із цих чисел утворити нову послідовність $f_1, f_2, ..., f_{n+k}$ так, щоб $f_1 \le f_2 \le ... \le f_{n+k}$.
11. Дано цілі числа $x_1, ..., x_n$. Упорядкувати дану послідовність за зростанням та знайти найбільше значення, що отримається в послідовності після вилучення з неї всіх членів зі значенням $\max(x_1, \ldots, x_n)$.
12. Дана матриця дійсних чисел (x_{ij}) порядку $m \times n$. Упорядкувати рядки матриці за не спаданням мінімальних елементів рядків.
13. Дана матриця дійсних чисел (x_{ij}) порядку $m \times n$. Упорядкувати рядки матриці за не зростанням максимальних елементів рядків.
14. Дана матриця дійсних чисел (x_{ij}) порядку $m \times n$. Упорядкувати стовпці матриці за не спаданням максимальних елементів стовпців.
15. Дана матриця дійсних чисел (x_{ij}) порядку $m \times n$. Упорядкувати стовпці матриці за не зростанням мінімальних елементів стовпців.

Підвищений рівень

1. Багаж пасажира характеризується кількістю та загальною вагою речей. Дано файл *bagaj*, який містить відомості про багаж декількох пасажирів. Відомості про багаж - це структури із двома полями: одне поле цілого типу (кількість речей) і друге – дійсного типу (вага в кілограмах). Упорядкувати відомості про багаж, що записані в файлі

bagaj, за не зростанням ваги багажу, а при однаковій вазі впорядкувати за не зростанням кількості речей.

2. Створити файл структур із полями: назва міста, кількість жителів, площа міста. Упорядкувати цей файл за не зростанням кількості жителів, а при однаковій кількості, упорядкувати за не спаданням площі.
3. Дано текстовий файл. Словом вважають групу символів між двома пробілами. Упорядкувати слова файлу за алфавітом (лексикографічно).
4. Упорядкувати список студентів за не зростанням середнього балу оцінок останньої сесії, а прізвища студентів, що мають однаковий середній бал, за алфавітом. Вивести результати на екран.
5. Створити файл структур із полями: прізвище, номер дому, заборгованість за квартиру. Упорядкувати цей файл за зростанням номеру дому, а в межах одного дому – за зростанням заборгованості.
6. Створити файл структур із полями: найменування товару, країна, що імпортує товар, та обсяг поставленої партії в штуках. Відсортувати файл за полем “обсяг поставленої партії в штуках” за зростанням. Визначити назву країни, яка імпортує найбільше товару.
7. Упорядкувати список студентів за місяцем народження, а прізвища студентів, що народилися в одному місяці, за алфавітом. Вивести результати на екран.
8. Створити файл структур із полями: прізвище, вік, зріст. За один перегляд файлу створити список, який складається з найвищих людей. Упорядкувати отриманий список за зростанням віку.
9. Створити файл структур із полями: прізвище, номер дому, заборгованість за квартиру. Визначити номер дому, жителі якого мають найбільшу сумарну заборгованість за квартиру. Упорядкувати дані про жителів цього дому за алфавітом прізвищ.
10. Дано файл *бібліотека* , який містить відомості про книги: прізвище автора, назва та рік видання. Упорядкувати даний файл за роком видання книг, а в межах одного року, за алфавітом прізвищ авторів.
11. Дано файл *асортимент*, який містить відомості про іграшки: назва іграшки (лялька, конструктор і т. д.), її вартість та вікові межі (наприклад, іграшка може бути призначена для дітей від 2 до 5 років). Упорядкувати даний файл за зростанням вікової межі та визначити, для якого віку є найбільше іграшок.
12. Дано файл, що містить номери телефонів співробітників закладу: прізвище співробітника, його ініціали та номер телефону. Упорядкувати даний файл за прізвищами співробітників (в алфавітному порядку). Вивести інформацію про співробітників, які користуються однаковими телефонами.
13. Створити файл структур із полями: назва міста, кількість жителів, площа міста. За один перегляд файлу створити список міст, які мають найбільшу площу. Упорядкувати цей список за не спаданням кількості жителів.
14. Створити файл структур із полями : прізвище, номер дому, заборгованість за квартиру. За один перегляд файлу створити список жителів, що проживають в домі №24. Упорядкувати отриманий список за спаданням заборгованості за квартиру.
15. Файл містить інформацію про студентів: прізвище, ім‘я та рік народження. Вивести на екран відомості про студентів з іменами, що починаються з літери “А”, упорядкувавши їх за роком народження.

Питання для самоконтролю

1. Що таке сортування даних?
2. У яких структурах даних можна проводити сортування?
3. Сформулюйте постановку задачі сортування.
4. Які методи сортування вам відомі?

5. Наведіть приклади алгоритмів сортування та оцінки їх ефективності.
6. Суть алгоритму сортування вставкою та його ефективність. У яких випадках доцільно його використовувати?
7. Суть алгоритму сортування вибором та його ефективність. У яких випадках доцільно його використовувати?
8. Суть алгоритму сортування обміном (бульбашкове сортування) та його ефективність. У яких випадках доцільно його використовувати?
9. Суть алгоритму швидкого сортування та його ефективність. У яких випадках доцільно його використовувати?
10. Суть алгоритму пірамідального сортування та його ефективність. У яких випадках доцільно його використовувати?
11. Суть алгоритму сортування методом злиття та його ефективність. У яких випадках доцільно його використовувати?

Практичне заняття № 14

*Тема: **Пошук***

Студент повинен знати: постановку задачі пошуку, області застосувань алгоритмів пошуку, алгоритми послідовного та бінарного пошуку та оцінки їх ефективності, застосування цих алгоритмів до розв'язування практичних задач.

Теоретичні відомості

Задача пошуку є фундаментальною в комбінаторних алгоритмах. У загальному її можна сформулювати так: «Дослідити множину T з метою відшукання елемента, що задовольняє деяку умову C». Ефективність алгоритму пошуку суттєво залежить від структури множини T.

Для отримання відповіді на питання «скільки існує способів…», «перерахуйте всі можливі…» та інше, зазвичай треба дослідити множини всіх можливих розв'язків, тобто виконати так званий вичерпний пошук. Існують загальні методи вичерпного пошуку – пошук із поверненням (бектрекінг) та метод решета. Пошук із поверненням використовується у широкому класі задач, що включають граматичний розбір, ігри та складання розкладів. На кожному кроці вичерпного пошуку робиться спроба розширити частковий розв'язок із метою отримання шуканого розв'язку. Метод решета є логічним доповненням вичерпного пошуку і полягає у виключенні тих об'єктів, що не є розв'язками. Цей метод корисний перш за все для теоретико-числових задач.

Важливим класом алгоритмів пошуку є алгоритми на графах, зокрема деревах. До них відносять алгоритми пошуку найкоротших шляхів між вершинами графа, пошуки в глибину і ширину, пошук максимального потоку в мережі, алгоритми обходу дерев.

Будемо розглядати структури даних, які є масивами або файлами. Для цих структур найпростішими алгоритмами пошуку є послідовний та бінарний (двійковий) пошук.

Послідовний пошук. Під послідовним пошуком розуміють дослідження елементів в тому порядку, у якому вони розташовані в послідовності (масиві, файлі). Нехай X[1], X[2],…, X[n] – масив із n елементів, і в цьому масиві треба знайти елемент Z, якщо він є. Алгоритм послідовного пошуку полягає у послідовному порівнянні елементів масиву X, починаючи з першого, із Z до тих пір, поки не знайдемо Z або не з'ясуємо, що Z у масиві немає. Доцільно доповнити масив елементом, який шукаємо, тобто присвоїти X[n+1]:=Z. Тоді алгоритм послідовного пошуку матиме вигляд:

```
X[n+1]:=Z;
i:=1;
while Z<>X[i] do i:=i+1;
if i<=n then {елемент X[i] співпадає із Z}
         else {Z не міститься в масиві}
```

До основних алгоритмів для послідовностей елементів відносять алгоритми пошуку мінімального чи максимального елемента. Такі алгоритми легко реалізувати через послідовний пошук, вибравши за початковий мінімальний (максимального) елемент перший елемент послідовності. Для прикладу, алгоритм пошуку мінімального елемента масиву X має вигляд.

```
K:=1;  {K – індекс мінімального елемента}
Min:=X[1];
for i:=1 to n do
  if Min>X[i] then
         begin Min:=X[i]; K:=i end;
```

Даний алгоритм знаходить мінімальний елемент та його індекс.

Ефективність алгоритму послідовного пошуку в гіршому випадку рівна O(n), бо треба переглядати всі елементи масиву, якщо шуканого елемента немає.

Бінарний пошук. Цей вид пошуку використовується до упорядкованих послідовностей. Нехай дано упорядкований масив X[1]≤X[2]≤ … ≤X[n], у якому треба знайти елемент Z. Спочатку Z порівнюється із середнім елементом масиву, тобто з елементом X[(n+1) div 2]. Якщо Z = X[(n+1) div 2], то пошук завершується. Якщо Z < X[(n+1) div 2], то розглядаємо ліву частину масиву, тобто X[1], X[2],…, X[(n+1) div 2 -1], а якщо Z > X[(n+1) div 2], то розглядаємо праву частину масиву, тобто X[(n+1) div 2 +1], X[(n+1) div 2 +2],…, X[n]. У вибраній частині шукаємо Z. Для цього Z знову порівнюємо із середнім елементом вибраної чистини масиву, і якщо знову Z не співпадає із середнім елементом, то визначаємо нову частину масиву, у якій будемо шукати Z. На кожному кроці алгоритму вибрана частина масиву для пошуку Z ділиться навпіл. Тому алгоритм припинить свою роботу, якщо буде знайдено Z або розмір частини масиву стане рівним 1. Фрагмент алгоритму бінарного пошуку для масиву X має вигляд.

```
J:=0;          {J – індекс елемента масиву X, який буде співпадати з Z}
L:=1;          {L – індекс лівого краю частини масиву, вибраної для пошуку}
R:=n;          {R - індекс правого краю частини масиву, вибраної для пошуку}
while L<=R do
begin
  m:= (L+R) div 2;          {m – індекс середнього елемента}
  if Z<X[m] then R:= m-1
        else if Z>X[m] then L:= m+1
              else begin J:= m; break end
end;
if J>0 then   {елемент Z співпадає з X[J]}
        else  {елемент Z не міститься в масиві}
```

Якщо початковий масив упорядкований за спаданням, то в цьому фрагменті алгоритму треба поміняти вибір лівого та правого країв масиву, у якому здійснюється пошук. Алгоритм бінарного пошуку дуже ефективний, і його часова складність $O(\log_2 n)$.

Приклад 1

Дано масив цілих чисел, які упорядковані за зростанням. Знайти в масиві задане число і вивести його номер.

Аналіз задачі

Розв’язування задачі полягає в застосуванні до масиву алгоритму бінарного пошуку для знаходження даного числа. Цей алгоритм реалізуємо у вигляді функції.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 10                    // Кількість елементів в масиві
// Функція реалізує алгоритм бінарного пошуку до упорядкованого за зростанням масиву.
// Через параметр b у функцію передається масив, n – кількість елементів масиву, z – число,
// пошук якого буде здійснюватися в масиві
int BinPoisk(int b[], int n, int z);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i;
        int x;                          // Число для пошуку в масиві
        int a[N];                       // Оголошення масиву
        printf ("Введіть %d цілих чисел упорядкованого за зростанням масиву\n", N);
        for (i=0; i<N; i++)             // Введення масиву з клавіатури
```

```
        scanf("%d", &a[i]);
    printf ("Виведення масиву\n");
    for (i=0; i<N; i++)
        printf ("%5d", a[i]);
    printf ("\n");
    printf ("Введіть ціле число для пошуку\n");
    scanf("%d", &x);            // Введення числа для пошуку в масиві
    i=BinPoisk(a, N, x);        // Пошук числа
    if (i>=0)
        printf ("Число %d е в масиві. Його індекс = %d\n", x, i);
    else
        printf ("Числа %d немае в масиві\n", x);
    system("pause");
    return 0;
}
//-----------------------------------------------------------------------------------------------------------
int BinPoisk(int b[], int n, int z)
{   int j=-1;               // Індекс елемента масиву, який буде співпадати з z
    int l=0;                // Індекс лівого краю частини масиву, вибраної для пошуку
    int r=n-1;              // Індекс лівого краю частини масиву, вибраної для пошуку
    int m;
    while (l<=r)
    {   m=(l+r)/2;          // Індекс середнього елемента частини масиву
        if (z<b[m])
            r=m-1;          // Зміна правого краю частини масиву, у якій
                            // буде здійснюватися пошук
        else
            if (z>b[m])
                l=m+1;      // Зміна лівого краю частини масиву, у якій
                            // буде здійснюватися пошук
            else            // Число z знайдене в масиві
            {   j=m;        // Запам'ятовуємо його індекс
                break;
            }
    }
    return j;
}
```

```
Введіть 10 цілих чисел упорядкованого за зростанням масиву
-6 -3 -1 0 3 6 8 9 13 20
Виведення масиву
   -6   -3   -1    0    3    6    8    9   13   20
Введіть ціле число для пошуку
12
Числа 12 немае в масиві
Для продолжения нажмите любую клавишу . . .
```

```
Введіть 10 цілих чисел упорядкованого за зростанням масиву
-6 -4 -1 1 4 7 9 10 15 20
Виведення масиву
   -6   -4   -1    1    4    7    9   10   15   20
Введіть ціле число для пошуку
9
Число 9 е в масиві. Його індекс = 6
Для продолжения нажмите любую клавишу . . .
```

Результати роботи програми.

Задачі

1. Створити файл дійсних чисел, який упорядкований за спаданням. Знайти в файлі задане число і вивести його номер, скориставшись алгоритмом бінарного пошуку.
2. Створити файл структур із полями: прізвище, ім'я, по батькові та рік народження. Вивести інформацію про наймолодших та найстарших людей.

☺ Індивідуальні завдання

Основний рівень

1. Дано упорядкований масив із N цілих чисел. Скласти програму пошуку в цьому масиві елементів, квадрат яких більший за 100. Вивести кількість цих елементів.
2. Скласти програму пошуку в упорядкованому масиві з N цілих чисел елемента, який у сумі з останнім елементом дорівнював би 65. Вивести відповідні повідомлення при наявності такого елемента в масиві або при його відсутності.
3. Дано файл цілих чисел $a_1,...,a_n$, які упорядковані за спаданням. Знайти серед цих чисел ті, що більші за середнє арифметичне суми всіх елементів. Вивести ці числа на екран.
4. Дано файл цілих чисел $a_1,...,a_n$, які упорядковані за неспаданням. Знайти серед цих чисел номер першого нуля.
5. Дано файл цілих чисел $a_1,...,a_n$, які упорядковані за незростанням. Знайти перший елемент цієї послідовності, який стоїть після двох поруч розташованих нулів. Вивести на екран цей елемент і його номер. Якщо ж такого елемента немає, то вивести відповідне повідомлення.
6. Дано файл дійсних чисел $a_1,...,a_n$, що упорядковані за спаданням. Знайти заданий елемент x. Якщо елемента x немає, то вставити його у файл так, щоб файл залишився упорядкованим за спаданням.
7. Дано файл дійсних чисел $a_1,...,a_n$, що упорядковані за зростанням. Знайти серед цих чисел елемент, рівний x^2 (x попередньо задати).
8. Дано файл дійсних чисел $a_1,...,a_n$, упорядкованих за зростанням. Знайти серед цих чисел перший елемент, синус якого більший за –0.89.
9. Дано цілі числа $a_1,...,a_n$, упорядковані за спаданням. Знайти номер першого елемента, який більший за даний елемент. Вивести знайдений елемент і його номер на екран.
10. Дано файл натуральних чисел $a_1,...,a_n$, упорядкованих за неспаданням. Підрахувати найбільшу кількість однакових елементів x (x попередньо задати).
11. Дано послідовність додатніх дійсних чисел $a_1,...,a_n$, упорядкованих за спаданням. Знайти серед елементів цієї послідовності ті, корінь квадратний із яких більший за середнє арифметичне першого і останнього елементів послідовності.
12. Дано файл дійсних чисел $a_1,...,a_n$ (n ввести з клавіатури), упорядкованих за зростанням. Вивести елементи, які більші за даний елемент x.
13. Дано файл дійсних чисел $a_1,...,a_n$ (n ввести з клавіатури), упорядкованих за неспаданням. Вивести на екран номер першого елемента, що безпосередньо розташований за двома однаковими елементами.
14. Дано файл дійсних чисел $a_1,...,a_n$ (n ввести з клавіатури), упорядкованих за спаданням. Знайти серед цих чисел елемент, рівний середньому арифметичному всіх елементів даного файлу.

15. Дано масив дійсних чисел $a_1,...,a_n$, упорядкованих за зростанням. Знайти номер першого елемента, який більший за a_1^3. Якщо такого елемента не існує, то вивести відповідне повідомлення.

Підвищений рівень

1. Нехай у бухгалтерії створено файл структур із полями: прізвище співробітника, рік народження, заробітна плата. Вивести прізвища тих співробітників, у яких заробітна плата більше 1650 грн.
2. Створити файл структур, полями якої є відомості про врожай у фермерському господарстві: назва посівної культури, рік посіву, урожайність (у відсотках). Вивести на екран відомості про врожай за 2015 рік.
3. Дано файл структур телефонної станції. Полями структури є прізвище абонента, його номер телефону. Знайти всі номери абонентів, прізвища яких починаються з приголосного.
4. Дано файл структур, полями яких є ідентифікаційний номер студента, розмір стипендії, курс, на якому він навчається. Вивести на екран відомості про стипендію студентів, які навчаються на 5-му курсі.
5. Дано файл структур, полями яких є номер випускного року, кількість випускників із золотими медалями, кількість випускників зі срібними медалями. Вивести на екран рік, у якому було випущено учнів із найбільшою кількістю золотих медалей.
6. Створити файл структур про відомості овочевої бази. Полями структури є назва овочу (буряк, морква, капуста, картопля), дата його надходження, кількість у тонах. Вивести на екран відомості про надходження і кількість моркви. Обчислити загальну кількість моркви.
7. Створити файл структур Укрпошти з полями: індекс, вид видання (журнал або газета) назва видання, вартість, прізвище замовника. Вивести відомості про прізвища замовників, які підписалися на газету "Голос України".
8. Створити файл структур із полями: назва вулиці, кількість будівель на цій вулиці, рік виникнення назви вулиці. Вивести на екран назви вулиць, які виникли після N-го року.
9. Створити файл структур про студентів деякої групи, куди занести прізвище і оцінки за останню зимову сесію. Знайти середній бал успішності студентів даної групи. Вивести прізвища студентів, успішність яких вище середньої.
10. Створити файл структур із полями: назва вулиці, назва будівлі на цій вулиці, рік зведення будівлі. Підрахувати і вивести на екран кількість будівель, які були побудовані за роки від N до M.
11. Створити файл структур про табір "Червоні вітрила" з полями: прізвище, вік дитини, стать, назва загону, номер зміни. Вивести на екран відомості про хлопчиків 2-ої зміни.
12. Створити файл структур із полями: прізвище, номер квартири, дата заселення. Вивести на екран відомості про мешканців, прізвища яких починаються з голосного.
13. Створити файл структур із полями: прізвище хворого, дата, з якої він потрапив у лікарню, дата одужання. Вивести прізвища хворих, які потрапили у лікарню пізніше певної дати.
14. Створити файл структур із полями: прізвище мешканця, номер квартири, дата заселення, кількість мешканців у квартирі. Знайти всі дані по мешканців, які заселилися першими.
15. Створити файл структур із полями: назва товару, ціна, кількість, дата постачання. Для комп'ютерної фірми повідомте, який товар користується найбільшим попитом (постачається найчастіше).

☝ Питання для самоконтролю

1. Сформулюйте постановку задачі пошуку.
2. При вирішенні яких задач використовуються алгоритми пошуку?
3. У чому суть алгоритму послідовного пошуку?
4. Яка різниця в алгоритмах пошуку заданого елемента і пошуку мінімального елемента?
5. Яка ефективність алгоритму послідовного пошуку? У яких випадках вона досягається?
6. У чому суть алгоритму бінарного пошуку? Чому він так називається?
7. Який метод побудови алгоритмів використаний при створенні алгоритму бінарного пошуку?
8. Яким алгоритмом пошуку краще користуватися в упорядкованому масиві?
9. Яка ефективність алгоритму бінарного пошуку? У яких випадках вона досягається?

Практичне заняття № 15

Тема: ***Однозв'язні лінійні списки.***

Студент повинен знати: поняття динамічної змінної, її оголошення та ініціалізацію, синтаксис оголошення типу елемента однозв'язного лінійного списку, створення та ініціалізацію однозв'язного лінійного списку, операції добавлення та вилучення елементів списку.

Теоретичні відомості

Динамічні змінні – це змінні, для яких оперативна пам'ять виділяється під час виконання програми, і які знищуються також під час виконання програми. Область пам'яті, у якій розташовуються ці змінні, називається **динамічною** або купою (від англ. heap – купа). Звичайні (статичні) змінні характеризуються тим, що їх значення зберігаються в ділянках оперативної пам'яті, які визначаються на етапі компіляції програми і не змінюються під час її виконання. Обсяг оперативної пам'яті, що необхідний для збереження значення динамічної змінної, компілятору невідомий. Тому доступ до значення такої змінної здійснюється за її *адресою*.

У мові С вказівники можуть адресувати ділянку пам'яті, зокрема ділянку динамічної пам'яті. Тому динамічні змінні реалізуються через вказівники і немає потреби в іменуванні таких змінних.

У заголовкових файлах **stdlib.h**, **alloc.h** містяться стандартні функції для роботи з динамічною пам'яттю. Основними є функції malloc(), calloc(), realloc(), free().

Основною функцією виділення динамічної пам'яті є функція

void * malloc(size_t size);

Параметр size задає обсяг (у байтах) ділянки динамічної пам'яті, яку треба виділити для даних. Функція повертає адресу першого байта виділеної ділянки або NULL, якщо виділити пам'ять не вдалося.

Іншою функцією, призначеною для виділення динамічної пам'яті, є

void * calloc(size_t num, size_t size);

Функція виділяє ділянку динамічної пам'яті для num елементів, кожний з яких має розмір size. Вона повертає адресу першого байта виділеної ділянки або NULL, якщо виділити пам'ять не вдалося.

Функція realloc() призначена для зміни обсягу ділянки динамічної пам'яті, попередньо виділеної функціями malloc() або calloc(). Прототип її такий:

void * realloc(void * ptr, size_t newsize);

тут ptr – вказівник на ділянку динамічної пам'яті, обсяг якої треба змінити (збільшити або зменшити); newsize – новий обсяг ділянки в байтах. У разі успішного завершення функція повертає адресу ділянки нового обсягу, а при невдачі – NULL. Стара ділянка пам'яті в останньому випадку не змінюється.

Функція

void free(void * ptr);

звільняє ділянку динамічної пам'яті, на початок якої вказує ptr.

Списком називають послідовність елементів одного типу. Прикладом списку є масив. Але розмір масиву визначається на етапі компіляції і залишається незмінним протягом усього часу виконання програми. Якщо у списку часто потрібно виконувати добавлення чи вилучення елементів, то доцільно реалізовувати список у вигляді динамічної структури даних. Динамічний список можна організувати, використовуючи можливості мови С для створення динамічних структур даних.

Зв'язний лінійний список – це сукупність однотипних елементів, які послідовно зв'язані між собою за допомогою вказівників. Кожен елемент списку, крім останнього, містить вказівник на наступний елемент. Доступ до першого елемента здійснюється за допомогою вказівника на нього, а доступ до кожного наступного елемента – з

використанням вказівника, який зберігається у попередньому елементі. Перший елемент списку називається його вершиною або головою.

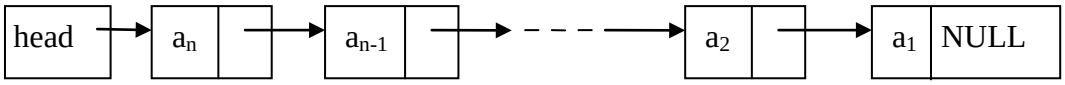
Над зв'язними лінійними списками виконують такі дії:

- добавлення нового елемента у початок списку;
- добавлення нового елемента в кінець списку;
- добавлення нового елемента між двома наявними елементами списку;
- вилучення елемента зі списку.

Зв'язні лінійні списки поділяються на такі різновиди:

- однозв'язний лінійний список;
- однозв'язний циклічний список;
- двозв'язний лінійний список;
- двозв'язний циклічний список;
- стек;
- черга.

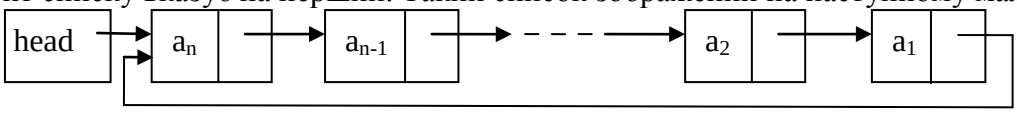
Однозв'язний лінійний список – це список, у якому попередній елемент вказує на наступний (попередній елемент містить вказівник на наступний). На малюнку 15.1 зображено такий список.



Мал. 15.1. Однозв'язний лінійний список.

Вказівник head показує на вершину списку, а елементи списку добавлялись до вершини, тобто список створювався справа наліво. Елементи a_1, a_2, …, a_n – це дані списку.

Однозв'язний циклічний список – це однозв'язний лінійний список, у якому останній елемент списку вказує на перший. Такий список зображений на наступному малюнку 15.2.



Мал. 15.2. Однозв'язний циклічний список.

Програмування однозв'язних лінійних списків починається з опису типу елемента списку. Елемент списку має тип структури, у якій повинні бути передбачені поля для даних і одне поле для вказівника. Доцільно структурному типу надавати власне ім'я. Тип такої структури можна визначити так

```
typedef struct elem
        { <тип даних> dani;
          struct elem * next;
        } element;                    // тип елемента списку
typedef element * ptr;                // тип вказівника на елемент списку
```

За допомого службового слова typedef ми визначили власний тип element для елементів списку. У ньому поле dani призначене для збереження даних елемента, а поле next є вказівником на наступний елемент списку. Зауважимо, що означення типу елемента списку є рекурсивним. Також введений тип ptr для вказівника на елемент списку.

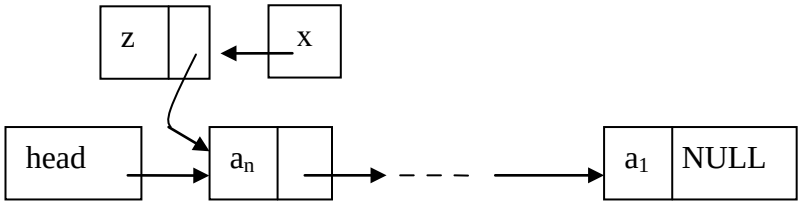
Після визначення типів у програмі можна оголошувати відповідні змінні. Оголосимо два вказівники

```
ptr head, current;
```

Вказівник head – це вказівник на вершину списку, а current – вказівник на поточний елемент.

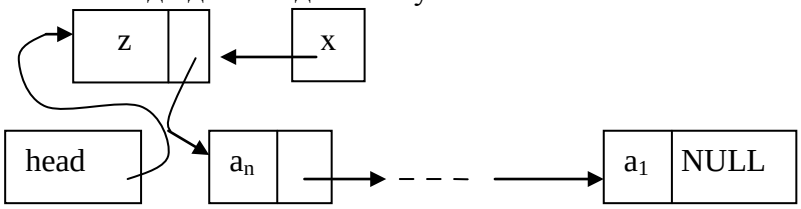
Операція добавлення нового елемента до списку полягає у тому, що спочатку в динамічній пам'яті створюється новий елемент, а потім він під'єднується до списку. Розглянемо різні варіанти цієї операції.

На малюнку 15.3 зображено створення нового елемента, що добавлятиметься до вершини списку.



Мал. 15.3. Створення елемента, на який вказує вказівник x.

Тепер цей елемент під'єднаний до списку.

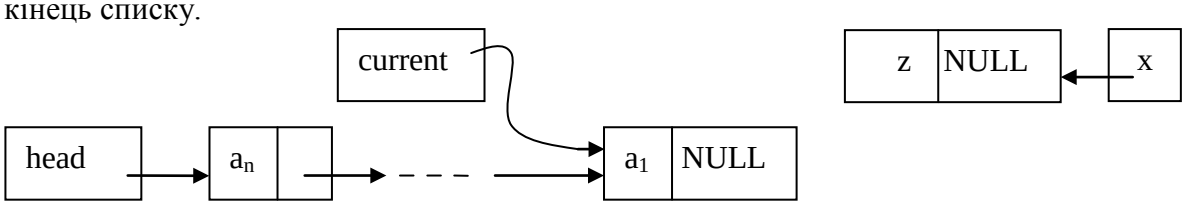


Мал. 15.4. Під'єднання елемента до списку.

Операцію добавлення елемента до вершини списку реалізує наступна функція, яка повертає вказівник на нову вершину списку.

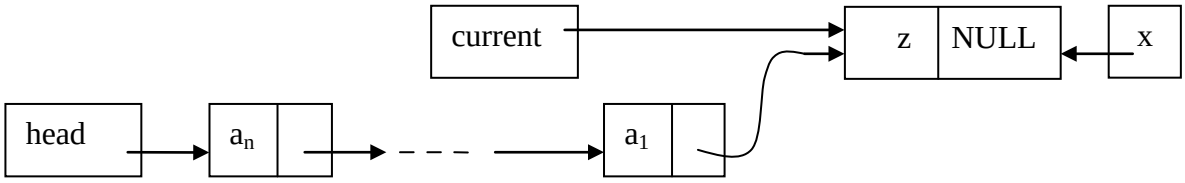
```
ptr InsertV(ptr head, <тип даних> z)
{       ptr x;                  // Оголошення допоміжного вказівника
        // Виділення динамічної пам'яті для елемента списку.
        // Адреса елемента запам'ятовується в x.
        x=(ptr) malloc(sizeof(element));
        // Заповнюємо поля структури
        x->dani = z;
        x->next = head;
        head = x;               // Вказівник на нову вершину списку
        return head;
}
```

Зобразимо на малюнку 15.5 створення нового елемента, що буде добавлятися у кінець списку.



Мал. 15.5. Створення елемента, на який вказує вказівник x.

Після добавлення елемента список матиме вигляд.



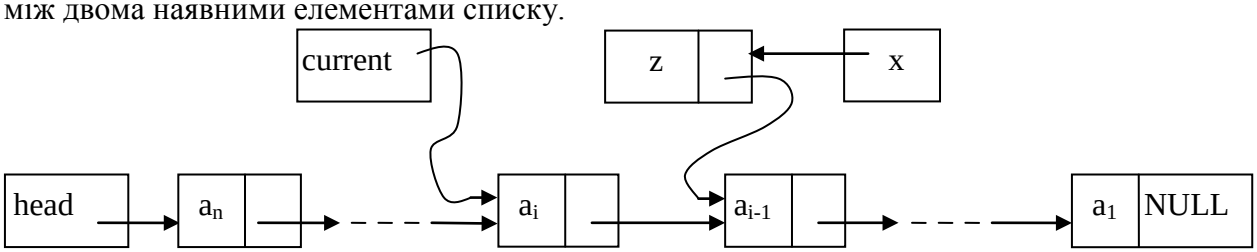
Мал. 15.6. Під'єднання елемента до списку.

Відповідна функція має вигляд, якщо вважати current вказівником на останній елемент списку.

```
ptr InsertK(ptr current, <тип даних> z)
{       ptr x;                  // Оголошення допоміжного вказівника
        // Виділення динамічної пам'яті для елемента списку.
        // Адреса елемента запам'ятовується в x.
        x=(ptr) malloc(sizeof(element));
        // Заповнюємо поля структури
        x->dani = z;
        x->next = NULL;
        current = x;            // Змінюємо вказівник, хоч це можна не робити
```

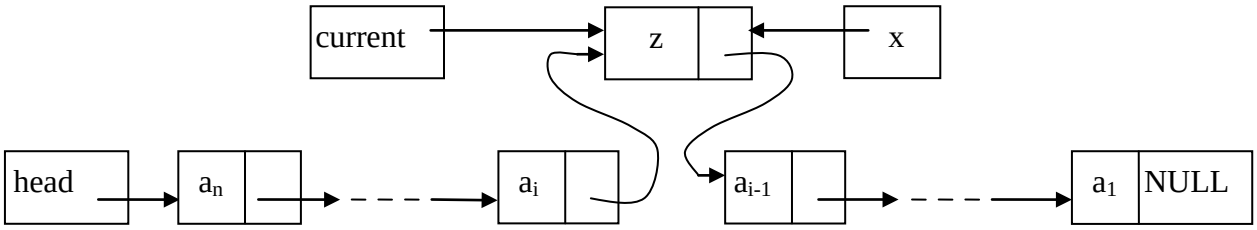
```
        return current;
}
```

Тепер зобразимо на малюнку 15.7 створення нового елемента, що буде добавлятися між двома наявними елементами списку.



Мал. 15.7. Створення елемента, на який вказує вказівник x.

Після добавлення елемента список матиме вигляд.

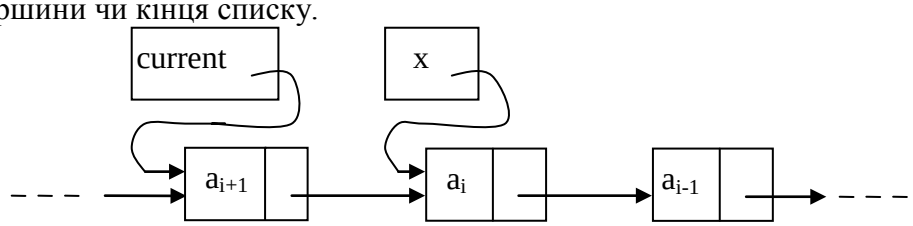


Мал. 15.8. Під'єднання елемента до списку.

Відповідна функція має вигляд, якщо вважати current вказівником на і-й елемент списку.

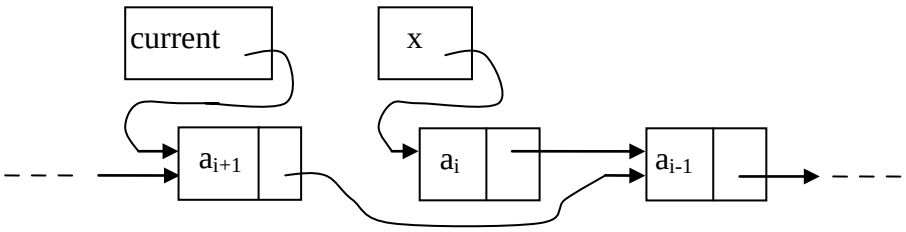
```
ptr InsertS(ptr current, <тип даних> z)
{       ptr x;                  // Оголошення допоміжного вказівника
        // Виділення динамічної пам'яті для елемента списку.
        // Адреса елемента запам'ятовується в x.
        x=(ptr) malloc(sizeof(element));
        // Заповнюємо поля структури
        x->dani = z;
        x->next = current->next;
        current->next = x;      // Під'єднання і-го елемента до створеного
        current = x;            // Змінюємо вказівник, хоч це можна не робити
        return current;
}
```

Аналогічно тому, як існують різні варіанти добавлення елемента до списку, також можна розглядати різні варіанти вилучення елемента. Розглянемо тільки вилучення елемента у середині списку. Внісши необхідні уточнення, можна отримати реалізацію вилучення із вершини чи кінця списку.



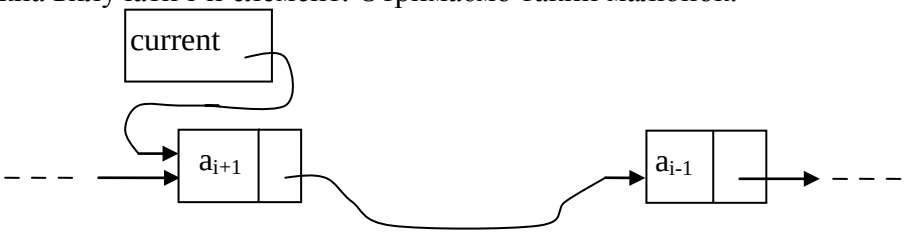
Мал. 15.9. Вказівник x вказує на елемент, який будемо вилучати.

На малюнку 15.9 зображені три елементи списку – (i+1)-й, i-й, (i-1)-й. Припустимо, що ми визначили (i+1)-й елемент, і нехай його вказівник current. Треба вилучити і-й елемент. Для цього спочатку треба визначити вказівник x на і-й елемент. Це робиться через присвоювання x=current->next.



Мал. 15.10. Обхід елемента, який будемо вилучати.

Наступним кроком операції вилучення є зміна посилання (i+1)-го елемента так, як показано на малюнку 15.10. Це робиться через присвоювання current->next=current->next->next. Тепер можна вилучати і-й елемент. Отримаємо такий малюнок.



Мал. 15.11. Список без вилученого елемента.

Операцію вилучення можна реалізувати наступною функцією.

```
ptr  DelElem(ptr current)
{       ptr x;                  // Оголошення допоміжного вказівника
        x = current->next;      // Вказівник X адресує і-й елемент
        current->next = current->next->next;        // Обхід і-го елемента
        free(x);                // Вилучення і-го елемента з динамічної пам'яті
        return current;
}
```

Цю функцію можна модифікувати для вилучення елемента з вершини чи кінця списку.

Для пошуку якогось елемента у списку чи дослідження всіх його елементів можна скористатися наступним фрагментом програми.

```
current := head;        //  Початкове значення поточного вказівника current
while  (current != NULL)
{       // Дослідити елемент списку
        current=current->next;      // Перехід до наступного елемента списку
}
```

Приклад 1

Створити список, який містить інформацію про студентів (поля: прізвище, чотири екзаменаційні оцінки за сесію). Знайти записи про студентів, які не склали сесію на «добре» та «відмінно». Вивести ці записи на екран та вилучити зі списку, тобто отримати список студентів, яким буде призначена стипендія.

Аналіз задачі

Створимо функцію InsertV() для добавлення елемента у вершину списку. Її використаємо при створенні списку. За допомогою функції DelSpisok() переглядатимемо створений список і будемо виводити інформацію про студентів, які отримали на екзамені 3 або 2. Також відповідні елементи вилучаємо зі списку. При вилученні використовуємо функцію DelElementV() для вилучення з вершини та функцію DelElementS() для вилучення з середини списку. Алгоритм вилучення такий:

- спочатку вилучаємо необхідні елементи з вершини до тих пір, поки не знайдемо студента, якому буде призначена стипендія (використовуємо функцію DelElementV());
- запам'ятовуємо вказівник на знайдену вершину списку;
- далі продовжуємо рух по решті списку і при необхідності вилучаємо елементи (використовується функція DelElementS()).

Отриманий новий список буде містити тільки дані про студентів, яким буде призначена стипендія. За допомогою функції DrucSpisokDel() виведемо цей список на екран і знищимо список, щоб звільнити динамічну пам’ять.

Зауважимо, що виведення списку здійснюється у зворотному порядку до його створення, бо нові елементи добавлялися у вершину, а не кінець списку.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
#include "stdlib.h"
// Оголошення типу element для елементів списку
typedef struct elem
        {       char prizv[20];                 // прізвище
                int ocinka[4];                  // чотири оцінки за сесію
                struct elem * next;     // вказівник на наступниу елемент, рекурсія в означенні
        }       element;
// Тип ptr – вказівник на елемент списку
typedef element * ptr;

// Функція добавляє новий елемент у вершину списку і повертає вказівник на нову
// вершину. Через параметр h у функцію передається вказівник на вершину списку,
// через s – прізвище студента, а через oc – масив оцінок за сесію.
ptr InsertV(ptr h, char * s, int oc[]);
// Функція виводить на екран інформацію про студентів.
// Через параметр h у функцію передається вказівник на вершину списку.
void DrucSpisok(ptr h);
// Функція вилучає елемент із вершини списку і повертає вказівник на нову вершину.
// Через параметр h у функцію передається вказівник на вершину списку.
ptr DelElementV(ptr h);
// Функція вилучає елемент із середини списку. Через параметр h у функцію передається
// вказівник на елемент, біля якого здійснюється вилучення. Функція повертає вказівник h.
ptr DelElementS(ptr h);
// Функція здійснює пошук у списку студентів, які отримали на екзамені 3 або 2.
// Відповідні елементи вилучаються зі списку.
// Через параметр h у функцію передається вказівник на вершину списку.
ptr DelSpisok(ptr h);
// Функція виводить на екран список і знищує його.
// Через параметр h у функцію передається вказівник на вершину списку.
void DrucSpisokDel(ptr h);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int j;
        element z;                              // Змінна для елемента списку
        // head – вказівник на вершину списку
        // current – вказівник на поточний елемент списку
        ptr head, current;
        head = NULL;                            // Початкова ініціалізація вказівника на список
        printf ("Введіть дані про студентів у список. Останне прізвище - #\n");
        fflush(stdin);                          // Очищення буфера введення
        printf ("Введіть прізвище:  ");
```

```
	gets(z.prizv);				// Вводимо прізвище
	while (strcmp(z.prizv, "#")!=0)		// Умова завершення циклу
	{	printf ("Введіть чотири оцінки:  ");
		for (j=0; j<4; j++)		// Вводимо чотири оцінки
			scanf("%d", &(z.ocinka[j]));
		head = InsertV(head, z.prizv, z.ocinka);	// Добавлення елемента у вершину
		fflush(stdin);			// Очищення буфера введення
		printf ("Введіть прізвище:  ");
		gets(z.prizv);			// Вводимо прізвище
	}
	printf ("Виведення створеного списку\n");
	DrucSpisok(head);			// Виведення списку на екран
	printf ("Виведення списку студентів, які отримали 3 або 2 не екзамені\n");
	// Виведення інформації про студентів, які отримали 3 або 2 на екзамені.
	// Вилучення відповідних елементів зі списку
	head = DelSpisok(head);
	if (head != NULL)
	{	printf ("Список студентів, яким буде призначена стипендія\n");
		DrucSpisokDel(head);
	}
	else
		printf ("Немае студентів, яким буде призначена стипендія\n");

	system("pause");
	return 0;
}
//---------------------------------------------------------------------------------------------------------------
ptr InsertV(ptr h, char * s, int oc[])
{	// Добавлення елемента у вершину списку
	int i;					// Параметр циклу
	ptr x;					// Оголошення вказівника на елемент списку
	x=(ptr) malloc(sizeof(element));	// Виділення динамічної пам’яті для елемента
	// Заповнення елемента
	strcpy(x->prizv, s);			// Заносимо прізвище
	for (i=0; i<4; i++)			// Заносимо оцінки за сесію
		x->ocinka[i]=oc[i];
	x->next = h;				// Адресуємо вказівник на вершину списку
	h = x;					// Змінюємо вершину
	return h;				// Повертаємо вершину
}
//---------------------------------------------------------------------------------------------------------------
void DrucSpisok(ptr h)
{	// Виведення даних списку на екран
	ptr x;		// Оголошення вказівника на елемент списку для руху по списку
	x=h;					// Спочатку він адресує вершину списку
	while (x != NULL)			// Рух по списку, поки не дійдемо до кінця
	{	// Виведення даних
		printf ("%25s%5d%5d%5d%5d\n", x->prizv, x->ocinka[0], x->ocinka[1],
							x->ocinka[2], x->ocinka[3]);
		x = x->next;		// Переадресовуємо вказівник на наступний елемент
	}
}
```

```
//-----------------------------------------------------------------------------------------------------------------
ptr DelElementV(ptr h)
{       // Вилучення елемента з вершини, h – вказівник на вершину
        ptr x;                          // Оголошення вказівника на елемент списку
        x = h;                          // Він дресує вершину
        h = h->next;                    // Зміна вказівника на вершину
        free(x);                        // Звільнення динамічної пам’яті
        return h;                       // Повертаємо вказівник на вершину
}
//-----------------------------------------------------------------------------------------------------------------
ptr DelElementS(ptr h)
{       // Вилучення елемента з середини списку,
        // h – вказівник на елемент, біля якого вилучаємо
        ptr x;                          // Оголошення вказівника на елемент списку
        x = h->next;                    // Він адресує елемент, який будемо вилучати
        h->next = h->next->next;        // Обхід елемента, що вилучаємо
        free(x);                        // Звільняємо динамічну пам’ять
        return h;
}
//-----------------------------------------------------------------------------------------------------------------
ptr DelSpisok(ptr h)
{       // Пошук студентів, які отримали 3 або 2 на екзамені.
        // Вилучення таких елементів зі списку, h – вказівник на вершину списку.
        ptr x;          // Оголошення вказівника на елемент списку для руху по списку
        // Вилучення із вершини
        while ((h->ocinka[0]<4) || (h->ocinka[1]<4) || (h->ocinka[2]<4) || (h->ocinka[3]<4))
        {       // Виведення даних
                printf ("%25s%5d%5d%5d%5d\n", h->prizv, h->ocinka[0], h->ocinka[1],
                                                        h->ocinka[2], h->ocinka[3]);
                h = DelElementV(h);         // Вилучення елемента
        }
        if (h != NULL)                  // Якшо список не порожній, продовжуємо пошук
        {       x = h;          // Поточний вказівник адресує вершину для руху по списку
                while (x->next != NULL)             // Поки не пройдемо список
                        if ((x->next->ocinka[0]<4) || (x->next->ocinka[1]<4) ||
                                (x->next->ocinka[2]<4) || (x->next->ocinka[3]<4))
                        {       Знайшли потрібний елемент, виводимо дані
                                printf ("%25s%5d%5d%5d%5d\n", x->next->prizv,
                                        x->next->ocinka[0], x->next->ocinka[1],
                                                x->next->ocinka[2], x->next->ocinka[3]);
                                x = DelElementS(x);         // Вилучаємо елемент
                        }
                        else                                // Елемент не відповідає умові
                                x = x->next;            // Перехід на наступний елемент
        }
        return h;                                   // Повертаємо вершину списку
}
//-----------------------------------------------------------------------------------------------------------------
void DrucSpisokDel(ptr h)
{       // Виводимо список на екран і знищуємо його, h – вказівник на вершину.
        while (h != NULL)                   // Поки не кінець списку
        {       // Виводимо дані
```

```
            printf ("%25s%5d%5d%5d%5d\n", h->prizv, h->ocinka[0], h->ocinka[1],
                                                    h->ocinka[2], h->ocinka[3]);
            h = DelElementV(h);          // Вилучаємо елемент з вершини списку
        }
}
```

```
Введіть дані про студентів у список. Останне прізвище - #
Введіть прізвище:  Rak
Введіть чотири оцінки:  3 4 3 4
Введіть прізвище:  Bober
Введіть чотири оцінки:  5 5 5 5
Введіть прізвище:  Ivanov
Введіть чотири оцінки:  2 2 3 2
Введіть прізвище:  Strixa
Введіть чотири оцінки:  4 5 4 4
Введіть прізвище:  Karp
Введіть чотири оцінки:  4 4 4 3
Введіть прізвище:  Rakov
Введіть чотири оцінки:  3 3 3 3
Введіть прізвище:  #
Виведення створеного списку
                   Rakov    3    3    3    3
                    Karp    4    4    4    3
                  Strixa    4    5    4    4
                  Ivanov    2    2    3    2
                   Bober    5    5    5    5
                     Rak    3    4    3    4
Виведення списку студентів, які отримали 3 або 2 не екзамені
                   Rakov    3    3    3    3
                    Karp    4    4    4    3
                  Ivanov    2    2    3    2
                     Rak    3    4    3    4
Список студентів, яким буде призначена стипендія
                  Strixa    4    5    4    4
                   Bober    5    5    5    5
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Задачі

1. Заданий текстовий файл, який складається зі слів. У кожному слові від 1 до 10 малих латинських букв, між словами пробіл, за останнім словом крапка. Надрукувати ці слова за алфавітом, вказавши для кожного з них число входжень у файл.
2. Задана не порожня послідовність натуральних числе, яка закінчується нулем. Вивести порядкові номери найменших чисел в послідовності.

☺ Індивідуальні завдання

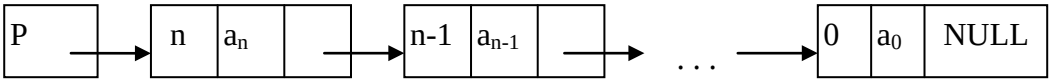
Основний рівень

1. Створити список, який містить інформацію про користувачів телефоном. (поля: прізвище, номер телефону, заборгованість). Знайти записи про клієнтів, борг яких перевищує деякий максимум. Вивести ці записи на екран та вилучити зі списку.
2. Створити список, який містить інформацію про користувачів водогоном (поля: прізвище, адреса, заборгованість). Перевірити список на наявність клієнтів, борг яких перевищує деякий мінімум. Вивести ці записи на екран та вилучити зі списку.
3. Створити список, який містить інформацію про пасажирів автобуса (поля: прізвище, номер місця, пункт призначення). Знайти інформацію про пасажирів, які прибули в певний пункт призначення. Вивести її на екран та вилучити відповідні записи зі списку.
4. Дано список слів. Написати програму, яка друкує текст із перших букв усіх слів списку.

5. Створити список, який містить інформацію про жителів міста. (поля: прізвище, адреса, вік). Перевірити список на наявність громадян, вік яких перевищує певний рівень. Вивести ці записи на екран та вилучити зі списку.
6. Створити список цілих чисел. Вивести на екран усі числа, які більші певного заданого числа, і вилучити їх зі списку.
7. Створити список цілих чисел. Вивести на екран усі числа, які менші певного заданого числа, і вилучити їх зі списку.
8. Створити список слів. Вивести на екран усі слова, які довші певної заданої довжини, і вилучити їх зі списку.
9. Створити список слів. Вивести на екран всі слова, які коротші певної заданої довжини, і вилучити їх зі списку.
10. Створити список слів. Замінити всі входження слова s1 на s2.
11. Створити список, який містить інформацію про користувачів телефоном. (поля: прізвище, номер телефону, заборгованість). Перевірити, чи є у списку дублікати записів про телефонних абонентів, та вилучити їх.
12. Перенести на початок списку його останній елемент.
13. Роздрукувати список у зворотному порядку.
14. Обчислити середнє арифметичне елементів списку цілих чисел і вилучити всі елементи більші нього.
15. Обчислити середнє арифметичне елементів списку цілих чисел і вилучити всі елементи менші нього.

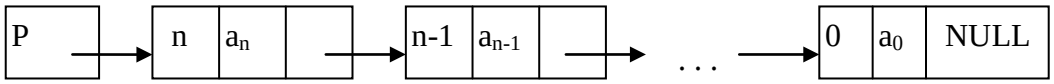
Підвищений рівень

1. Дано три списки L, L1, L2 цілих чисел. Створити процедуру, яка у списку L заміняє перше входження списку L1 (якщо таке є) на список L2.
2. Створити програму, яка зі списків слів L1, L2 формує новий список L, заносячи до нього по одному разу ті слова, що містяться хоча б в одному зі списків L1 і L2.
3. Подвоїти входження кожного елемента списку. Елементи, які входять до списку більше двох разів, вилучити.
4. Файл містить дані: прізвище, адреса, телефон, зарплата. Відсортувати його за зростанням зарплати, використавши список. Прізвищами людей із однаковою зарплатою відсортувати в алфавітному порядку.
5. Файл містить дані: прізвище, адреса, телефон. Відсортувати прізвищами в алфавітному порядку, використавши список, а при наявності однакових прізвищ відсортувати адреси теж в алфавітному порядку. Здійснити пошук абонента за номером телефону.
6. Дано два списки цілих чисел. Утворити один список, відсортований за зростанням.
7. Дано два списки, елементами яких є слова. Утворити один список, відсортований від “Я” до “А”.
8. Дано два списки цілих чисел. Створити список, елементами якого є числа, які входять одночасно до обох списків.
9. Дано два списки цілих чисел. Створити список, елементами якого є числа, які входять до першого списку і не входять до другого.
10. (Лічилка) n дітей розташовуються по кругу. Почавши відлік від першого, вилучають кожного k-го, замикаючи круг після кожного вилучення. Написати програму, яка за даними n і k виводить номери дітей у тому порядку, за якими вони вилучалися із круга.
11. Многочлен $P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ можна представити у вигляді списку.



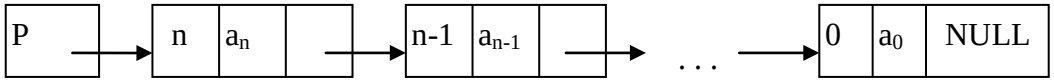
Якщо $a_i = 0$, то відповідний запис не включається до списку. Описати тип даних, що відповідає такому представленню многочленів, і створити логічну функцію *дорівнює*(P,Q), яка перевіряє рівність многочленів P і Q.

12. Многочлен $P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ можна представити у вигляді списку.



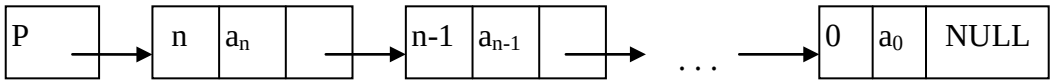
Якщо $a_i = 0$, то відповідний запис не включається до списку. Описати тип даних, що відповідає такому представленню многочленів, і створити логічну функцію *знач*(P,X), яка обчислює значення многочлена P в цілочисловій точці X.

13. Многочлен $P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ можна представити у вигляді списку.



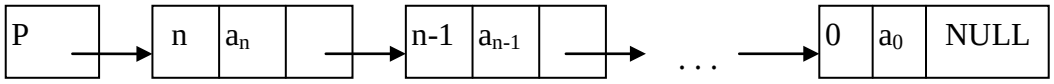
Якщо $a_i = 0$, то відповідний запис не включається до списку. Описати тип даних, що відповідає такому представленню многочленів, і створити функцію *сум*(P,Q,R), яка обчислює многочлен R, який дорівнює сумі многочленів P і Q.

14. Многочлен $P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ можна представити у вигляді списку.



Якщо $a_i = 0$, то відповідний запис не включається до списку. Описати тип даних, що відповідає такому представленню многочленів, і створити функцію *добуток*(P,Q,R), яка обчислює многочлен R, який дорівнює добутку многочленів P і Q.

15. Многочлен $P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ можна представити у вигляді списку.



Якщо $a_i = 0$, то відповідний запис не включається до списку. Описати тип даних, що відповідає такому представленню многочленів, і створити функцію *діф*(P,Q), яка обчислює многочлен Q, що є похідною многочлена P.

Питання для самоконтролю

1. Що таке динамічна пам’ять? Для чого вона служить?
2. Вкажіть функції для роботи з динамічною пам’яттю.
3. Що таке список?
4. Назвіть різновиди лінійних списків.
5. Дайте означення однозв’язного лінійного (циклічного) списку.
6. Напишіть функцію створення списку.
7. Які основні операції над списками ви знаєте?

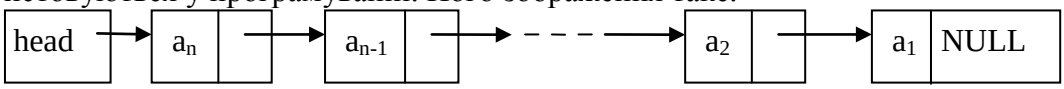
Практичне заняття № 16

Тема: ***Стеки***

Студент повинен знати: поняття стеку та синтаксис оголошення типу його елементів, створення та ініціалізацію стеку, операції добавлення та вилучення елементів.

Теоретичні відомості

Стек – однозв'язний лінійний список, у якому елементи добавляються та вилучаються лише з його вершини, тобто з початку списку. Стеки також інколи називають «магазинами», а в англомовній літературі для позначення стеків ще використовується абревіатура LIFO (last-in-first-out – останнім прийшов – першим вийшов). Стек широко використовуються у програмуванні. Його зображення таке.



Вказівник head показує на вершину стеку, і тільки через цей вказівник здійснюється добавлення та вилучення елементів. Функція добавлення елемента наводилася у занятті 15, і вона має вигляд.

```
ptr InsertV(ptr head, <тип даних> z)
{       ptr x;                  // Оголошення допоміжного вказівника
        // Виділення динамічної пам'яті для елемента списку.
        // Адреса елемента запам'ятовується в x.
        x=(ptr) malloc(sizeof(element));
        // Заповнюємо поля структури
        x->dani = z;
        x->next = head;
        head = x;               // Вказівник на нову вершину списку
        return head;
}
```

Аналогічна функція вилучення елемента має вигляд.

```
ptr DelElementV(ptr head)
{       // Вилучення елемента з вершини, head – вказівник на вершину
        ptr x;                  // Оголошення вказівника на елемент списку
        x = head;                       // Він дресує вершину
        head = head->next;              // Зміна вказівника на вершину
        free(x);                // Звільнення динамічної пам'яті
        return head;            // Повертаємо вказівник на вершину
}
```

Стеки можна реалізовувати також за допомогою масивів. Крім того, вони дозволяють перетворити рекурсивну підпрограму у нерекурсивну.

Приклад 1

Створити файл структур із полями: прізвище, стать, рік народження. За один перегляд файлу вивести інформацію про найстарших чоловіків.

Аналіз задачі

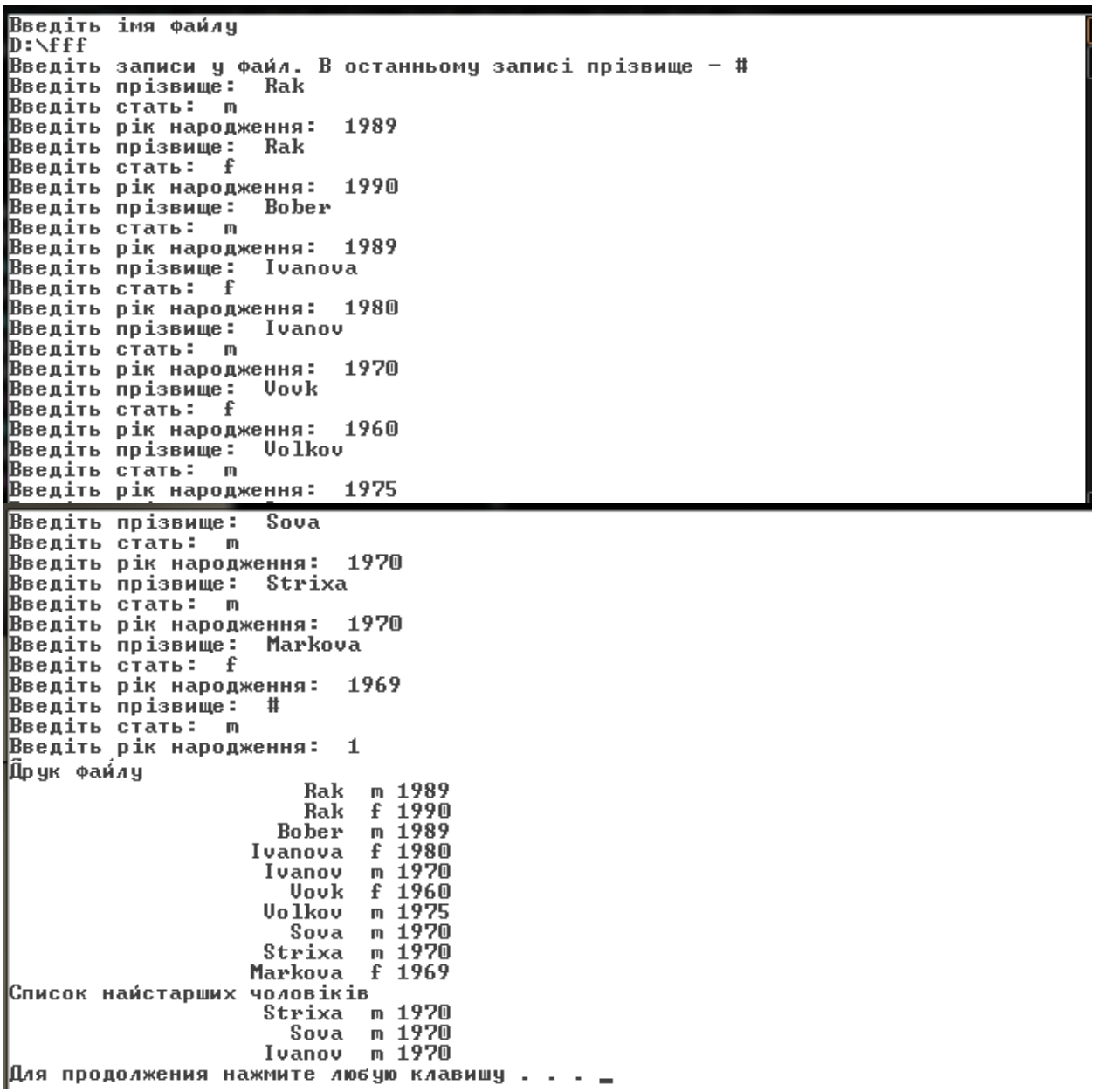
Спочатку треба створити файл структур. Доцільно це організувати у вигляді функції. Також створимо функцію для виведення даних файлу на екран. Пошук найстарших чоловіків будемо здійснювати послідовно переглядаючи файл від початку до кінця. Спочатку занесемо до стеку перший запис про чоловіка, якого вважатимемо найстаршим. Переглядаючи далі файл, якщо зустрічаємо дані про чоловіка того ж віку, то теж заносимо їх до стеку. Якщо знайшли дані про молодшого чоловіка, то їх ігноруємо (не заносимо до стеку). Якщо знайшли дані про старшого чоловіка, то повністю очищуємо

стек і заносимо до нього нові дані. Працюючи за таким алгоритмом, після перегляду файлу у стеку будуть дані про найстарших чоловіків. Виведення даних зі стеку буде здійснюватися в порядку зворотному до порядку їх розташування у файлі. Якщо потрібно було б зберігати при виведенні порядок розташування даних про найстарших чоловіків у файлі, то доцільно використати чергу замість стеку.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
#include "stdlib.h"
// Оголошення структурного типу anketa для даних файлу
typedef struct inf
        {       char prizv[20];             // прізвище
                char sex;                   // стать
                int rik;                    // рік народження
        }       anketa;
// Оголошення структурного типу element для елементів стеку
typedef struct elem
        {       anketa dani;                // дані про людей
                struct elem * next;         // вказівник
        }       element;
// Визначення структурного типу ptr для вказівника на елемент стеку
typedef element * ptr;
// Функція для заповнення полів структури типу anketa. Через вказівник p заповнена
// структура передається із функції в програму
void StvorenStruct(anketa *p);
// Функція створює файл структур. Через вказівник p передається інформація про файл
void StvorenFile(FILE *p);
// Функція виводить на екран вміст  файлу структур.
void DrucFile(FILE *p);
// Функція добавляє елемент у вершину стеку. Параметр head – це вказівник на вершину
// стеку, z – вказівник на структуру, дані якої заносяться в елемент, що добавляється.
// Функція повертає новий вказівник на вершину стеку.
ptr InsertV(ptr head, anketa * z);
// Функція вилучає елемент із вершини стеку. Параметр head – це вказівник на вершину
// Функція повертає новий вказівник на вершину.
ptr DelElementV(ptr head);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int r=3000;             // Початкове значення року народження найстаршого чоловіка
        char filename[120];             // Змінна для імені файлу
        printf ("Введіть імя файлу\n");
        gets(filename);                 // Введення імені файлу з клавіатури
        FILE * fp;                      // Оголошення вказівника на потік
        fp=fopen(filename, "wb");       // Відкриття двійкового файлу для запису
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        StvorenFile(fp);                // Створення двійкового файлу структур
```

```
        fp=fopen(filename, "rb");      // Відкриття двійкового файлу для читання
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        printf ("Друк файлу\n");
        DrucFile(fp);                  // Виведення файлу на екран
        rewind(fp);                    // Встановлення поточного вказівника на початок файлу
        anketa x;                      // Оголошення змінної структурного типу для файлу
        ptr head = NULL;               // Початкове значення вказівника на вершину стеку
        fread(&x, sizeof(anketa), 1, fp);       // Читаємо структуру з файлу
        while (!feof(fp))              // Поки не кінець файлу
        {       if (x.sex == 'm')      // Якщо прочитали дані про чоловіка
                {       if (x.rik == r)        // Він є найстаршим на даний момент
                                head = InsertV(head, &x);     // Заносимо його дані в стек
                        else
                                if (x.rik < r)    // Знайдений ще старший чоловік
                                {       if (head == NULL)     // Якщо стек порожній
                                                head = InsertV(head, &x);     // Заносимо дані
                                                                старшого чоловіка в стек
                                        else            // Стек не порожній
                                                {       // Очищуємо стек
                                                while (head != NULL)
                                                        head = DelElementV(head);
                                                // Заносимо дані старшого чоловіка в
                                                        порожній стек
                                                head = InsertV(head, &x);
                                                }
                                // Запам’ятовуємо рік народження найстаршого чоловіка
                                        r = x.rik;
                                }
                }
                fread(&x, sizeof(anketa), 1, fp);       // Знову читаємо структуру з файлу
        }
        fclose(fp);                    // Закриття файлу
        printf ("Список найстарших чоловіків\n");
        // Виводимо дані про найстарших чоловіків і знищуємо стек
        while (head != NULL)
        {       printf ("%25s%3c%5d\n", head->dani.prizv, head->dani.sex, head->dani.rik);
                head = DelElementV(head);
        }
        system("pause");
        return 0;
}
//-----------------------------------------------------------------------------------------------------------------
void StvorenStruct(anketa *p)
{       // Заповнення структури типу anketa
        fflush(stdin);                 // Очищення буфера введення
        printf ("Введіть прізвище:  ");
        gets(p->prizv);                // Введення прізвища з клавіатури
        printf ("Введіть стать:  ");
        fflush(stdin);                 // Очищення буфера введення
```

```
        scanf("%c", &(p->sex));        // Введення статі людини: ‘m’ – чоловік, ‘f’ - жінка
        printf ("Введіть рік народження:  ");
        scanf("%d", &(p->rik));        // Введення року народження
}
//-------------------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{       printf ("Введіть записи у файл. В останньому записі прізвище - #\n");
        anketa x;                      // Оголошення структурної змінної
        StvorenStruct(&x);             // Заповнення структури
        while (strcmp(x.prizv, "#")!=0)
        {       fwrite(&x, sizeof(anketa), 1, p);      // Запис структури в файл
                StvorenStruct(&x);                     // Заповнення чергової структури
        }
        fclose(p);                                     // Закриття файлу
}
//-------------------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{       anketa x;                                      // Оголошення структурної змінної
        fread(&x, sizeof(anketa), 1, p);               // Читання структури з файлу
        while (!feof(p))                               // Поки не кінець файлу
        {       printf ("%25s%3c%5d\n", x.prizv, x.sex, x.rik);      // Виведення на екран
                fread(&x, sizeof(anketa), 1, p);       // Читання наступної структури з файлу
        }
}
//-------------------------------------------------------------------------------------------------------------------
ptr InsertV(ptr head, anketa * z)
{       // Добавлення елемента у вершину стеку, head – вказівник на вершину
        ptr x;                                 // Оголошення допоміжного вказівника
        x = (ptr) malloc(sizeof(element));     // Виділення динамічної пам’яті для нового
                                               елемента списку, адресу елемента запа’ятовуємо в x
        // Занесення даних в новий елемент списку
        strcpy(x->dani.prizv, z->prizv);
        x->dani.sex = z->sex;
        x->dani.rik = z->rik;
        x->next = head;
        head = x;                              // Змінюємо вказівник на вершину списку
        return head;                           // Повертаємо цей вказівник
}
//-------------------------------------------------------------------------------------------------------------------
ptr DelElementV(ptr head)
{       // Вилучення елемента із вершини стеку, head – вказівник на вершину
        ptr x;                         // Оголошення допоміжного вказівника
        x = head;                      // Запам’ятовуємо елемент, який вилучатимемо
        head = head->next;             // Обхід цього елемента
        free(x);                       // Вилучення його з динамічної пам’яті
        return head;                   // Повертаємо новий вказівник на вершину стеку
}
```



Результат роботи програми.

Приклад 2

Дано деякий текст, що закінчується крапкою (у сам текст крапка не входить). Визначити, чи є цей текст правильним записом «формули»

<формула>::= <терм> | (<формула> <знак> <формула>)
<знак>::= + | - | *
<терм>::= <ім'я> | <ціле>
<ім'я>::= <буква> | <ім'я> <буква> | <ім'я> <цифра>
<ціле>::= <цифра> | <ціле> <цифра>
<буква>::= a | b | c | d | e | f | g
<цифра>::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Аналіз задачі

Згідно означення найпростішими формулами є терми, які є іменами або цілими. Імена задаються словами (ідентифікаторами), першим символом яких є буква, а наступні символи – букви або цифри. Довжина імена нічим не обмежена, однак конкретне ім'я є скінченним словом. Шаблон для імен можна представити регулярним виразом

<буква> (<буква> | <цифра>)*,

а шаблон для цілих – виразом <цифра> (<цифра>)*.

Розпізнавання цих виразів легко запрограмувати. Зауважимо, однако, що цілим є таке число 001 або 0000 і т.п.

Складні формули утворюються з термів за допомого трьох операцій +, - , * та круглих дужок. Круглі дужки визначають порядок виконання операцій. В означенні формули три операції рівноправні, тобто мають однаковий приорітет. Загалом, фомула складається з термів та символів +, -, *, (,). Довжина термів не регламентована, а довжина символів 1.

Для розпізнавання синтаксису формули використаємо стек, даними якого будуть символи +, -, *, (,), t. Символ t будемо записувати в стек, якщо успішно розпізнали терм. Сам терм заносити в стек немає сенсу, бо ми тільки розпізнаємо формулу, а не обчислюємо її. Алгоритм розпізнавання формули такий:

1. Читаємо перший символ формули;
2. Нехай це не відкрита дужка ‘(‘. Тоді формула має вигляд терму. Якщо в процесі подальшого читання символів формули ми отримали тільки терм, то формула записана правильно, інакше ні. Алгоритм завершує роботу.
3. Нехай першим символом є ‘(‘. Тоді формула повинна мати складну структуру. Створюємо стек та заносимо в нього в порядку розпізнавання символи символи +, -, *, (, t. до тих пір, поки не прочитаємо закриту дужку ‘)’. Тоді вилучаємо з вершини стека чотири елементи. При правильному записі формули ці елементи повинні містити символи t (терм), знак (одна з операцій +, -, *), t (терм), закрита дужка ‘)’. Заносимо в стек символ t і продовжуємо аналіз. Кожний раз коли буде зустрічатися закрита дужка ми повторюватимемо вилучення зі стеку. Якщо формула записана вірно, то після читання останньої закритої дужки ми вилучимо зі стеку всі елементи та припиняємо роботу алгоритму.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
#include "stdlib.h"
// Визначення типу element для елементів стеку
typedef struct elem
        {       char sym;                       // символ
                struct elem * next;             // вказівник
        }       element;
// Визначення типу ptr для вказівника на елемент стеку
typedef element * ptr;
// Функція добавляє елемент у вершину стеку та повертає вказівник на нову вершину.
// Через параметр head передається вказівник на вершину стеку, а через параметр z –
// символ формули.
ptr InsertV(ptr head, char z);
// Функція вилучає елемент із вершини стеку. Через параметр head передається вказівник
// на вершину стеку.
ptr DelElementV(ptr head);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int n;                          // Довжина формули
        int i=0, j;                     // i – індекс символа формули
        int flag;               // Прапорець, який інформує чи правильно записана формула
        ptr head = NULL;                // Вказівник на вершину стека
```

```
char formula[120];              // Змінна для формули, яка є рядком
printf ("Введіть формулу\n");
gets(formula);          // Введення формули з клавіатури. Кінець формули - крапка
n=strlen(formula);              // Визначення довжини формули
// Аназізуємо перший (нульовий) символ формули
if (((formula[i]>='a')&&(formula[i]<='g'))||((formula[i]>='0')&&(formula[i]<='9')))
{       // Перший символ є буквою або цифрою. Формула є термом
        if ((formula[i]>='a')&&(formula[i]<='g'))    // Перший символ буква
        {       // Розпізнаємо в циклі терм, який є іменем
                while ((i < n-1) &&(((formula[i]>='a')&&(formula[i]<='g'))
                                        ||((formula[i]>='0')&&(formula[i]<='9'))))
                        i++;    // Збільшуємо індекс – рух по формулі
                // Цикл завершить роботу, якщо зустрівся символ, що не
                // повинен входить в ім’я, або завершили читати формулу
                if ((i==n-1)&&(formula[i]=='.'))      // Прочитали всю формулу і
                                                        вона є термом
                        flag=1;         // Успішно розпізнали формулу
                else
                        flag=0;         // Формула записана невірно
        }
        else            // Перший символ цифра
        {       // Розпізнаємо в циклі терм, який є цілим
                while ((i < n-1)&&(formula[i]>='0')&&(formula[i]<='9'))
                        i++;
                if ((i==n-1)&&(formula[i]=='.'))      // Прочитали всю формулу і
                                                            вона є термом
                        flag=1;         // Успішно розпізнали формулу
                else
                        flag=0;         // Формула записана невірно
        }
}
else    // Перший символ не є буквою або цифрою. Формула має складну будову
{       while (i<n-1)           // Цикл для руху по формулі
        {// і-й символ формули є буквою. Він є першим символом імені
        if ((formula[i]>='a')&&(formula[i]<='g'))
        {       // Цикл для розпізнавання імені
                while ((i < n-1) &&(((formula[i]>='a')&&(formula[i]<='g'))
                                        ||((formula[i]>='0')&&(formula[i]<='9'))))
                        i++;
                if ((i<n-1)&&((formula[i]=='+')||(formula[i]=='-')
                                        ||(formula[i]=='*')||(formula[i]==')')))
                        // Якщо після імені зустрівся один із символів - +,-,*,)
                        head=InsertV(head, 't');        // Заносимо 't' в стек
                else    // Після імені зустрівся інший символ
                {       flag=0;         // Формула записана невірно
                        i=n;            // Завершуємо розпізнавання
                }
        }
        else            // і-й символ формули не є буквою
                if ((formula[i]>='0')&&(formula[i]<='9'))    // Він є цифрою
                {       // Цикл для розпізнавання цілого
                        while ((i < n-1)&&(formula[i]>='0')&&(formula[i]<='9'))
```

```
                i++;
            if ((i<n-1)&&((formula[i]=='+')||(formula[i]=='-')
                                ||(formula[i]=='*')||(formula[i]==')')))
            // Якщо після цілого зустрівся один із символів - +,-,*,)
                head=InsertV(head, 't');        // Заносимо 't' в стек
            else    // Після цілого зустрівся інший символ
            {       flag=0;         // Формула записана невірно
                i=n;                // Завершуємо розпізнавання
            }
    }
    else    // і-й символ формули не є буквою або цифрою
            if ((formula[i]=='+')||(formula[i]=='-')
                                ||(formula[i]=='*')||(formula[i]=='('))
            {       // і-й символ формули є + або – або * або (
                // Заносимо його в стек
                head=InsertV(head, formula[i]);
                i++;            // Індек наступного символа
            }
            else
            if (formula[i]==')')    // і-й символ формули є )
            {// Перевіряємо, чи в стеку, починаючи з вершини, містяться
            // символи в такому порядку - 't', один із символів +,-,*,
            // знову 't', нарешті '('.
            if ((head->sym=='t')&&((head->next->sym=='+')||
                        (head->next->sym=='-')||(head->next->sym=='*'))
                        &&(head->next->next->sym=='t')&&
                                    (head->next->next->next->sym=='('))
            {// Перевірка успішна. Вилучаємо зі стека читири елементи
                for (j=0; j<4; j++)
                    head= DelElementV(head);
                if (head==NULL)     // Якщо стек порожній
                    if (formula[i+1]=='.') // Кінець формули
                    {flag=1;        // Успішно розпізнали формулу
                    i=n;            // Завершуємо розпізнавання
                    }
                    else    // Стек порожній, а ще не кінець формули
                    {flag=0;        // Формула записана невірно
                    i=n;            // Завершуємо розпізнавання
                    }
                else    // Стек не порожній
                {if (i<n-1)     // Не кінець формули
                    {// Заносимо в стек символ 't'
                    head=InsertV(head, 't');
                    i++;    // Індек наступного символа
                    }
                    else    // Кінець формули, а стек не порожній
                        {flag=0;        // Формула записана невірно
                        i=n;            // Завершуємо розпізнавання
                        }
                }
            }
            else    // Перевірка не успішна
```

```
                                        {       flag=0;         // Формула записана невірно
                                                i=n;            // Завершуємо розпізнавання
                                        }
                                }
                                else    // і-й символ формули не належить алфавіту
                                        {       flag=0;         // Формула записана невірно
                                                i=n;            // Завершуємо розпізнавання
                                        }
                        }       // Кінець while (i<n-1)
                }
                if (flag)       // Виведення результату аналізу формули
                        puts(«Формула записана вірно»);
                else
                        puts(«Формула записана не вірно»);
                system(«pause»);
                return 0;
}
//-------------------------------------------------------------------------------------------------------------------
ptr InsertV(ptr head, char z)
{       ptr x;
        x = (ptr) malloc(sizeof(element));
        x->sym = z;
        x->next = head;
        head = x;
        return head;
}
//-------------------------------------------------------------------------------------------------------------------
ptr DelElementV(ptr head)
{       ptr x;
        x = head;
        head = head->next;
        free(x);
        return head;
}
```

```
Введіть формулу
(d2a+((43-abcd)*(((e1fcg*02)+aaa)-923))).
Формула записана вірно
Для продолжения нажмите любую клавишу . . .
```

```
Введіть формулу
123abc.
Формула записана не вірно
Для продолжения нажмите любую клавишу . . .
```

```
Введіть формулу
((x1-x2)+x3).
Формула записана не вірно
Для продолжения нажмите любую клавишу . . .
```

```
Введіть формулу
(a1-a2)*b.
Формула записана не вірно
Для продолжения нажмите любую клавишу . . .
```

```
Введіть формулу
((((12*13)*(a-b))*c)-2).
Формула записана вірно
Для продолжения нажмите любую клавишу . . .
```

Результати роботи програми.

Приклад 3

Створити файл цілих чисел. За один перегляд файлу знайти найменше число та вивести його порядкові номери у файлі. Використати стек, який реалізувати у вигляді масиву.

Аналіз задачі

Створення файлу цілих чисел реалізуємо у вигляді функції. Також створимо функцію для виведення даних файлу на екран. Пошук найменшого числа будемо здійснювати послідовно переглядаючи файл від початку до кінця. У стек будемо заносити порядкові номери цього числа. Спочатку занесемо до стеку номер першого числа, яке вважатимемо найменшим. Переглядаючи далі файл, якщо зустрічаємо таке саме число, то теж заносимо його номер до стеку. Якщо чергове число виявиться меншим за поточне найменше, то повністю очищуємо стек і заносимо до нього нові дані. Працюючи за таким алгоритмом, після перегляду файлу у стеку будуть номери найменшого числа.

Для стека виділимо масив, розмір якого повинний бути достатнім для збереження стека. Вказівник на вершину стека назвемо head. Його значенням є індекс елемента масиву. Стек порожній, якщо head == -1, бо індексація елементів масиву починається з 0.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 10                    // Розмір масиву для стека
// Функція створює двійковий файл цілих чисел.
void StvorenFile(FILE *p);
// Функція виводить на екран вміст двійкового файлу цілих чисел.
void DrucFile(FILE *p);
// Функція добавляє елемент у вершину стека і повертає індекс нової вершини. Через
// параметр mas передається масив (стек), n – розмір масиву, p – індекс вершини, z –
// порядковий номер найменшого елемента у файлі.
int AddStac(int mas[], int n, int p, int z);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i;
        int head = -1;                  // Індекс вершини стека
        int stac[N];                    // Масив для стека
        char filename[120];             // Змінна для імені файлу
        printf ("Введіть імя файлу f\n");
        gets(filename);                 // Введення з клавіатури імені файлу
        FILE * fp;
        fp=fopen(filename, "wb");       // Відкриття двійкового файлу для запису
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        StvorenFile(fp);                // Створення двійкового файлу
        fp=fopen(filename, "rb");       // Відкриття двійкового файлу для читання
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        printf ("Друк файлу\n");
        DrucFile(fp);                   // Виведення вмісту файлу на екран
```

```
        rewind(fp);                     // Встановлення поточного вказівника на початок файлу
        int x;
        int min;                                        // Змінна для найменшого числа
        int index = -1;                                 // Змінна для номерів чисел у файлі
        fread(&x, sizeof(int), 1, fp);                  // Читаємо з файлу перше число
        min = x;                                        // Вважаємо його найменшим
        index = 0;                                      // Його порядковий номер
        head = AddStac(stac, N, head, index);           // Заносимо цей номер в стек
        fread(&x, sizeof(int), 1, fp);                  // Читаємо з файлу друге число
        while (!feof(fp))                               // Поки не кінець файлу
        {       index++;                                // Збільшуємо номер числа
                if (x < min)            // Якщо число менше найменшого поточного числа
                {       min = x;                        // Нове найменше число
                        head = -1;                      // Очищення стеку
                        head = AddStac(stac, N, head, index);       // Заносимо номер нового
                                                                    найменшого числа в стек
                }
                else
                        if (x == min)                   // Якщо число рівне айменшому
                        head = AddStac(stac, N, head, index);// Заносимо його номер в стек
                fread(&x, sizeof(int), 1, fp);          // Читаємо з файлу наступне число
        }
        fclose(fp);                                     // Закриваємо файл
        printf ("Найменше число файлу = %d\n", min);
        printf ("Його індекси у файлі\n");              // Виведення номерів найменшого числа
        for (i=0; i<=head; i++)
                printf ("%d  ", stac[i]);
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{       // Організовуємо цикл для введення цілих чисел з клавіатури і запису їх у файл,
        // що пов’язаний з вказівником p. Якщо введене число 999, то цикл припиняє
        // роботу. Це число не заноситься у файл.
        printf ("Введіть числа у файл. Останне число - 999\n");
        int x;
        scanf("%d", &x);        // Введення цілого числа з клавіатури
        while (x!=999)
        {       fwrite(&x, sizeof(int), 1, p);          // Записуємо число у файл
                scanf("%d", &x);                        // Вводимо чергове ціле число
        }
        fclose(p);                                      // Закриття файлу
}
//-------------------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{       int x;
        fread(&x, sizeof(int), 1, p);                   // Читаємо число x з файлу
        while (!feof(p))                                // Поки не кінець файлу
        {       printf("%d   ", x);                     // Виводимо число x на екран
                fread(&x, sizeof(int), 1, p);           // Читаємо наступне число x з файлу
        }
```

```
        printf("\n");
}
//-------------------------------------------------------------------------------------------------------------
int AddStac(int mas[], int n, int p, int z)
{       p++;                            // Новий індекс вершини стеку
        if (p>=n)                       // Перевірка повноти стеку
                printf("Стек повний. Добавити елемент не можна\n");
        else
                mas[p] = z;             // Заносимо порядковий номер найменшого числа в стек
        return p;                       // Повертаємо індекс вершини стеку
}
```

```
Введіть імя файлу f
D:\fff
Введіть числа у файл. Останне число - 999
4 90 -5 45 2 12 -3 99 -1 0 1 12
6 -5 -5 -5 12 23 -3 -4 -2 56 90
999
Друк файлу
4   90   -5   45   2   12   -3   99   -1   0   1   12   6   -5   -5   -5   12
23   -3   -4   -2   56   90
Найменше число файлу = -5
Його індекси у файлі
2  13  14  15  Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Задачі

1. Виконати завдання прикладу 1, реалізувавши стек у вигляді масиву.
2. Дано деякий текст, що закінчується крапкою (у сам текст крапка не входить). Цей текст є правильним правильним записом формули, значення якої треба обчислити. Визначення формули наступне

   <формула>::= <ціле_число> | (<формула> <знак> <формула>)

   <знак>::= + | - | *

☺ Індивідуальні завдання

Основний рівень

1. Створити файл дійсних чисел. За один перегляд файлу вивести номери чисел, що мають найбільшу величину.
2. Створити файл записів із полями: прізвище, номер телефону, заборгованість. За один перегляд файлу вивести прізвища абонентів і номери телефонів, у яких заборгованість найбільша.
3. Послідовність цілих чисел вводиться з клавіатури. Не зберігаючи всієї послідовності у пам’яті, вивести порядкові номери найбільших чисел послідовності.
4. Створити текстовий файл. За один перегляд файлу створити новий файл, у якому будуть знаходитися всі слова, довжина яких буде найбільшою.
5. Створити файл структур із полями: прізвище, номер телефону. За один перегляд файлу знайти номери телефонів абонентів, які мають найкоротші прізвища.
6. Дано текстовий файл. Між сусідніми словами кома, за останнім словом крапка. За один перегляд файлу надрукувати всі слова найбільшої довжини.
7. Створити файл структур із полями: назва товару, ціна товару. За один перегляд файлу вивести назви всіх найдорожчих товарів.
8. Створити текстовий файл. За один перегляд файлу створити новий файл, у якому будуть знаходитися всі слова, довжина яких буде найменшою.
9. Створити файл структур із полями: прізвище, номер телефону. За один перегляд файлу знайти номери телефонів абонентів, які мають найдовші прізвища.

10. Створити файл структур із полями: назва товару, ціна товару. За один перегляд файлу вивести назви всіх найдешевших товарів.
11. Створити файл структур із полями: прізвище, номер телефону. За один перегляд файлу вивести прізвища абонентів, сума цифр номера телефону яких найменша.
12. Дано файл цілих чисел. За один перегляд файлу вивести всі числа, сума цифр яких найбільша.
13. Дано текстовий файл. Вивести вміст файлу на екран, друкуючи символи кожного рядка у зворотному порядку.
14. Створити файл структур із полями: прізвище, вік. За один перегляд файлу вивести прізвища найстарших людей.
15. Дано текстовий файл. Між сусідніми словами кома, за останнім словом крапка. За один перегляд файлу надрукувати всі слова найменшої довжини.

Підвищений рівень

1. Створити файл структур із полями: прізвище, номер телефону, заборгованість. За один перегляд файлу вивести прізвища абонентів за алфавітом і номери телефонів, у яких заборгованість найбільша.
2. Дано рядок. Перевірити, чи є він правильним записом формули такого виду:
   <формула>::=<терм>|<терм>+<формула>|<терм>-<формула>
   <терм>::=<ім'я>|(<формула>)|[<формула>]|{<формула>}
   <ім'я>::=x|y|z
3. Створити файл структур із полями: прізвище студента, оцінки за екзамени. За один перегляд файлу вивести за алфавітом прізвища студентів, які набрали найбільшу кількість балів.
4. Дана послідовність цілих чисел. За один перегляд послідовності вивести всі числа, сума цифр яких найменша, за зростанням.
5. Створити файл структур із полями: прізвище абітурієнта, оцінки за вступні екзамени. За один перегляд файлу вивести за алфавітом прізвищ абітурієнтів, які набрали найменшу кількість балів.
6. Створити файл дійсних чисел. За один перегляд файлу підрахувати кількість найменших і найбільших чисел та вивести їх порядкові номери.
7. У рядку записана без помилок формула такого виду:
   <формула>::=<цифра>|M(<формула>,<формула>)|m(<формула>,<формула>)
   <цифра>::=0|1|2|3|4|5|6|7|8|9
   де M означає функцію max, а m – min. Обчислити (як ціле число) значення даної формули (наприклад, M(5,m(6,8)) = 6).
8. Створити файл структур із полями: прізвище, номер телефону, заборгованість. За один перегляд файлу вивести інформацію про абонентів із найдовшими прізвищами, упорядкувавши її за спаданням заборгованості.
9. У рядку записаний без помилок логічний вираз (ЛВ) у такій формі:
   <ЛВ>::=true|false|($\neg$<ЛВ>)|(<ЛВ>$\wedge$<ЛВ>)|(<ЛВ>$\vee$<ЛВ>)
   де знаки $\neg$, $\wedge$ і $\vee$ позначають відповідно заперечення, кон'юнкцію і диз'юнкцію. Обчислити значення цього виразу.
10. У рядку записаний текст, що збалансований за круглими дужками:
    <текст>::=<пусто>|<елемент><текст>
    <елемент>::=<буква>|(<текст>)
    Треба для кожної пари дужок, що відкриваються і закриваються, надрукувати номери їх позицій у тексті, упорядкувавши пари номерів за зростанням номерів позицій дужок, що закриваються. Наприклад, для тексту A+(45-F(X)*(B-C)) треба надрукувати: 8 10; 12 16; 3 17.
11. У рядку записаний текст, що збалансований за круглими дужками:
    <текст>::=<пусто>|<елемент><текст>

<елемент>::=<буква>|(<текст>)

Треба для кожної пари дужок, що відкриваються і закриваються, надрукувати номери їх позицій у тексті, упорядкувавши пари номерів за зростанням номерів позицій дужок, що відкриваються. Наприклад, для тексту A+(45-F(X)*(B-C)) треба надрукувати: 3 17; 8 10; 12 16.

12. Створити файл структур із полями: прізвище, номер телефону, заборгованість. За один перегляд файлу вивести інформацію про абонентів, у яких заборгованість найменша, за спаданням номеру телефону.
13. Дано текстовий файл. Між сусідніми словами кома, за останнім словом крапка. За один перегляд файлу надрукувати за алфавітом усі слова найменшої довжини.
14. Дано файл цілих чисел. За один перегляд файлу вивести всі числа, сума цифр яких найбільша, за спаданням.
15. У рядку записана без помилок формула такого виду:

<формула>::=<цифра>|(<формула>+<формула>)|(<формула>-<формула>)

<цифра>::=0|1|2|3|4|5|6|7|8|9

Обчислити (як ціле число) значення даної формули (наприклад, ((2-(4+6))+5) = -3).

♖ Додаткові задачі

1. *Ханойська вежа*. Є три стержні A,B,C та n дисків різного розміру, пронумерованих від 1 до n у порядку зростання розмірів. Спочатку всі диски знаходяться на стержні A: найбільший внизу, найменший зверху. Потрібно перенести усі диски зі стержня A на стержень C, дотримуючись вимог: переносити можна лише по одному диску, більший диск не можна класти на менший. Стержень B допоміжний. Використовуючи стек, створити нерекурсивний варіант алгоритму, що виводить виконувані перенесення.
2. *Пошук у глибину*. Задано зв'язний неорієнтований граф із n вершин списками суміжностей. Пошук у глибину дозволяє відвідати всі вершини графа та отримати стовбурне дерево цього графа. Створити програму рекурсивного та не рекурсивного варіантів цього алгоритму. Також програма повинна виводити стовбурне дерево графа.
3. Дано наступне означення виразу:

<вираз>::=<терм>|<терм><знак><вираз>

<знак>::= +|-

<терм>::=<множник>|<множник>*<терм>

<множник>::=<число>|<змінна>|(<вираз>)|<множник>^<число>

<число>::=<цифра>

<змінна>::=<буква>

де знак ^ означає операцію піднесення до степеня.

У математиці прийнята інфіксна форма запису арифметичних виразів із можливим використанням дужок. Для бінарної операції інфіксна форма виглядає як A○B, де ○ – символ бінарної операції, A і B – операнди. Постфіксною формою запису (польський запис) виразу A○B називають запис, у якому знак операції розташований за операндами: AB○. Якщо знак операції розташовується перед операндами, тобто ○AB, то отримуємо префіксну форму запису. Постфіксна та префіксна форми не містять дужок і є зручними для обчислення значення виразу на комп'ютері. Приклади форм запису наведені у таблиці.

Інфіксна форма	Постфіксна форма	Префіксна форма
a-b	ab-	-ab
a*b+c	ab*c+	+*abc
a*(b+c)	abc+*	*a+bc
a+b^c^d*e	abc^d^e*+	+a*^^bcde

а) Дано рядок, у якому записаний числовий вираз (згідно вищезазначеного означення виразу, і вираз містить тільки числа) у постфіксній формі. Написати програму

обчислення значення цього виразу. Використати наступний алгоритм обчислення. Вираз переглядається зліва направо. Якщо зустрічається операнд (число), то його значення (як ціле) заноситься до стеку. Якщо зустрічається знак операції, то зі стеку вилучаються два останніх елементи (це операнди даної операції), над ними виконується операція, і її результат записується до стеку. Зрештою у стеку залишиться тільки одне число – значення всього виразу.

b) Дано рядок, у якому записаний числовий вираз (згідно вищезазначеного означення виразу, і вираз містить тільки числа) у префіксній формі. Написати програму обчислення значення цього виразу. Скористатися алгоритмом із пункту a), тільки вираз треба переглядати справа наліво.

c) Дано рядок inf, у якому записаний вираз (згідно вищезазначеного означення) у звичайній (інфіксній) формі. Написати програму, яка перекладає вираз у постфіксну форму і записує його у новий рядок postf. Використати наступний алгоритм перекладу. Вираз переглядається зліва направо. Якщо зустрічається операнд (число або змінна), то він зразу записується в рядок postf. Якщо зустрічається дужка, що відкривається, то вона заноситься до стеку. Якщо зустрічається дужка, що закривається, то зі стеку переносяться у рядок postf всі операції, які містяться у стеку до найближчої відкриваючої дужки. Операції переносяться у рядок postf у тому порядку, як вони вилучаються зі стеку. Найближча відкриваюча дужка теж вилучається зі стеку, але дужки у рядок postf не записуються. Якщо ж при перегляді рядка inf зустрічається операція, то зі стеку вилучаються (до найближчої дужки, що є у стеку, або дна стеку) знаки операцій, старшинство яких більше або рівне старшинства даної операції, і вони записуються у рядок postf. Після цього дана операція заноситься до стеку. Алгоритм припиняє роботу, коли початковий рядок inf переглянутий повністю, і стек порожній. Якщо ж стек не порожній, то треба вилучити з нього всі операції (дужок у стеку вже немає) та записати їх у порядку вилучення у рядок postf.

☝ Питання для самоконтролю

1. Що таке динамічна пам’ять? Для чого вона служить?
2. Що таке вказівник?
3. Що таке стек?
4. У якому порядку виводяться елементи стеку?
5. Запишіть функцію вилучення елемента зі стеку.
6. Запишіть функцію добавлення елемента до стеку.
7. Основні операції над стеком.
8. Який механізм роботи стеку?
9. До якого елементу може бути застосована операція вилучення зі стеку?
10. Призначення функції malloc() та її параметри.
11. Який зв’язок між рекурсивними алгоритмами та організацією роботи зі стеком?
12. Як перетворити рекурсивний алгоритм у нерекурсивний?
13. Запропонуйте реалізацію стеку у вигляді масиву.
14. Наведіть приклади задач, у яких необхідно використовувати стек.

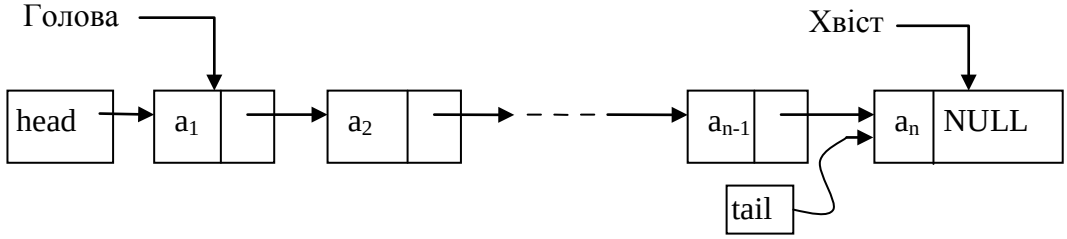
Практичне заняття № 17

Тема: ***Черги***

Студент повинен знати: поняття черги та синтаксис оголошення типу її елементів, створення та ініціалізацію черги, операції добавлення та вилучення елементів.

Теоретичні відомості

Черга – однозв'язний лінійний список, у якому елементи добавляються до кінця списку, а вилучаються лише з вершини, тобто з початку списку. В англомовній літературі для позначення черг використовується абревіатура FIFO (first-in-first-out – першим прийшов – першим вийшов). Черга дуже широко використовуються у програмуванні. Її зображення таке:



Черга має «голову» і «хвіст». У хвіст добавляються елементи, і тому необхідний вказівник tail, щоб мати можливість виконувати операцію добавлення. Вказівник head показує на голову черги і через нього вилучаються елементи. Функція добавлення елемента у хвіст наведена нижче. Параметр phead є вказівником на head, а ptail – вказівником на ptail. Через ці параметри відповідні вказівники передаються у функцію та із функції.

```
void InsertCherga(ptr * phead, ptr * ptail, <тип даних> z)
{       ptr x;                  // Оголошення допоміжного вказівника
        // Виділення динамічної пам'яті для елемента черги.
        // Адреса елемента запам'ятовується в x.
        x=(ptr) malloc(sizeof(element));
        // Заповнюємо поля структури
        x->dani = z;
        x->next = NULL;
        if (*phead == NULL)             // Черга порожня
        {       // Створюємо чергу з одного елемента
                *phead = x;
                *ptail = x;
        }
        else                                    // Черга не порожня
        {       (*ptail) -> next = x;           // Під'єднуємо елемент у хвіст черги
                *ptail = x;                     // Змінюємо вказівник на хвіст черги
        }
}
```

Функція вилучення елемента з черги має такий самий вигляд, як і для стеку.

```
ptr DelElementV(ptr head)
{       // Вилучення елемента з вершини, head – вказівник на вершину
        ptr x;                          // Оголошення вказівника на елемент списку
        x = head;                               // Він дресує вершину
        head = head->next;                      // Зміна вказівника на вершину
        free(x);                        // Звільнення динамічної пам'яті
        return head;                    // Повертаємо вказівник на вершину
}
```

Черги можна реалізовувати також за допомогою масивів. Такий масив є замкненим кільцем.

Приклад 1

Дано текстовий файл. За один перегляд файлу вивести спочатку слова, які складаються не більше ніж із трьох символів, а потім решту, зберігаючи порядок розташування слів у файлі.

Аналіз задачі

Спочатку створимо текстовий файл, використавши для цього функцію. Інформацію у файл записуємо рядками. Процес створення файлу триватиме до тих пір, поки не буде введений порожній рядок. Цей рядок не буде занесений до файлу.

Тепер будемо послідовно переглядати файл від початку до кінця. З файлу читаємо черговий рядок і будемо його розбивати на слова. Вважаємо, що роздільними символами між словами можуть бути пробіл, крапка, кома, двокрапка, крапка з комою, тире, знак оклику, знак питання. За допомогою функції strtok() будемо виокремлювати слова з рядка. Якщо отрималося слово довжиною не більше 3 символів, то зразу виводимо його на екран. Якщо ж слово довше, то заносимо його до черги. Після завершення перегляду файлу знищуємо чергу, виводячи її слова на екран.

```
#include "stdafx.h"
#include "windows.h"
#include "string.h"
// Визначення типу element для елементів черги
typedef struct elem
        {       char slovo[20];                 // слово
                struct elem * next;             // вказівник
        }       element;
// Визначення типу ptr для вказівника на елемент черги.
typedef element * ptr;
// Функція створює тексовий файл
void StvorenFile(FILE *p);
// Функція виводить тексовий файл на екран.
void DrucFile(FILE *p);
// Функція добавляє елемент до черги. Параметри: phead – подвійний вказівник на
// вершину черги, ptail – подвійний вказівник на хвіст черги, z – вказівник на слово.
void InsertCherga(ptr * phead, ptr * ptail, char * z);
// Функція вилучає елемент з черги і повертає новий вказівник на вершину.
ptr DelElementV(ptr head);

int _tmain(int argc, _TCHAR* argv[])
{       int k;
        char filename[120];             // Змінна для імені файлу
        printf ("Vvedit imja faylu\n");
        gets(filename);                 // Введення імені файлу з клавіатури
        FILE * fp;
        fp=fopen(filename, "w");        // Відкриття файлу для запису
        if (fp==NULL)
        {       puts("file ne vidkryvsja");
                exit(0);
        }
        StvorenFile(fp);                // Створення текстового файлу
        fp=fopen(filename, "r");        // Відкриття файлу для читання
```

```
        if (fp==NULL)
        {       puts("file ne vidkryvsja");
                exit(0);
        }
        printf ("Druk file\n");
        DrucFile(fp);                                   // Виведення файлу на екран
        rewind(fp);                     // Встановлення поточного вказівника на початок файлу
        ptr head = NULL;                                // Вказівник на вершину черги
        ptr tail = NULL;                                // Вказівник на хвіст черги
        char rydok[128];                                // Змінна для рядка
        char s[9]=" .,:;-!?";                           // Рядок роздільних символів між словами
        char *s1;
        puts("Slova do 3 symvoliv");
        fgets(rydok, 127, fp);                          // Читаємо рядок з файлу
        while (!feof(fp))                               // Поки не кінець файлу
        {       k = strlen(rydok);                      // Обчислюємо довжину рядка
                rydok[k-1] = '\0';                      // Замінюємо символ ‘\n’ на ‘\0’
                s1=strtok(rydok, s);                    // Виділяємо слово s1 в рядку
                while (s1!=NULL)                        // Розбиття рядка на слова
                {       if (strlen(s1) < 4)
                                printf ("%s  ", s1);    // Виводимо коротке слово на екран
                        else
                                InsertCherga(&head, &tail, s1);  // Довге слово заносимо в чергу
                        s1=strtok(NULL, s);
                }
                fgets(rydok, 127, fp);                  // Читаємо з файлу наступний рядок
        }
        fclose(fp);                                     // Закриваємо файл
        printf ("\n");
        puts("Reshta sliv failu");
        // Виводимо довгі слова файлу на екран і знищуємо чергу
        while (head != NULL)
        {       printf ("%s  ", head->slovo);
                head = DelElementV(head);
        }
        printf ("\n");
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{       // Створення текстового файлу через введення рядків з клавіатури
        printf ("Vvedit rjadky u file. Osnanniy rjadok - enter\n");
        char s[128];
        fgets(s, 127, stdin);                   // Вводимо рядок з клавіатури
        while (strcmp(s, "\n")!=0)
        {       fputs(s, p);                    Записуємо його в файл
                fgets(s, 127, stdin);           // Вводимо наступний рядок з клавіатури
        }
        fclose(p);
}
//-------------------------------------------------------------------------------------------------------------
```

```
void DrucFile(FILE *p)
{       char s[128];
        fgets(s, 127, p);                       // Читаємо рядок з файлу
        while (!feof(p))
        {       fputs(s, stdout);               // Виводимо його на екран
                fgets(s, 127, p);               // Читаємо наступний рядок з файлу
        }
}
//-------------------------------------------------------------------------------------------------------------------
void InsertCherga(ptr * phead, ptr * ptail, char * z)
{       // Добавлення елемента в чергу
        ptr x;
        x=(ptr) malloc(sizeof(element));        // Виділяємо для нового елемента пам'ять
        // Заповнюємо поля елемента
        strcpy(x->slovo, z);
        x->next = NULL;
        if (*phead == NULL)                     // Якщо черга порожня
                {       // Створюємо чергу з одного елемента
                        *phead = x;
                        *ptail = x;
                }
                else                            // Черга не порожня
                {       (*ptail) -> next = x;   // Під'єднємо новий елемент у хвіст черги
                        *ptail = x;             // Змінюємо вказівник на хвіст
                }
}
//-------------------------------------------------------------------------------------------------------------------
ptr DelElementV(ptr head)
{       // Вилучення елемента з вершини черги
        ptr x;
        x = head;
        head = head->next;
        free(x);
        return head;
}
```

```
Uvedit imja faylu
D:\fff
Uvedit rjadky u file. Osnanniy rjadok - enter
Раз, два, три, чотири. Вийшли хлопці із квартири?
Стали бігати та гратись. Ура!
Весна на дворі. А ви повірили?

Druk file
Раз, два, три, чотири. Вийшли хлопці із квартири?
Стали бігати та гратись. Ура!
Весна на дворі. А ви повірили?
Slova do 3 symvoliv
Раз  два  три  із  та  Ура  на  А  ви
Reshta sliv failu
чотири  Вийшли  хлопці  квартири  Стали  бігати  гратись  Весна  дворі  повірили

Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 2

Створити файл цілих чисел. За один перегляд файлу знайти найбільше число та вивести за зростанням його порядкові номери у файлі. Використати чергу, яку реалізувати у вигляді масиву.

Аналіз задачі

Створення файлу цілих чисел реалізуємо у вигляді функції. Також створимо функцію для виведення даних файлу на екран. Пошук найбільшого числа будемо здійснювати послідовно переглядаючи файл від початку до кінця. У чергу будемо заносити порядкові номери цього числа. Спочатку занесемо до черги номер першого числа, яке вважатимемо найбільшим. Переглядаючи далі файл, якщо зустрічаємо таке саме число, то теж заносимо його номер до черги. Якщо чергове число виявиться меншим за поточне найбільше, то повністю очищуємо чергу і заносимо до неї нові дані. Працюючи за таким алгоритмом, після перегляду файлу у черзі будуть номери найбільшого числа у порядку зростання.

Для черги виділимо масив, розмір якого повинний бути достатнім для збереження черги. Вказівник на вершину черги назвемо head, а на хвіст – tail. Їхніми значеннями є індекси елементів масиву. Черга порожня, якщо head == -1 та tail == -1, бо індексація елементів масиву починається з 0. Масив для черги є аналогом кільця, бо в процесі роботи черга буде зсуватися по масиву.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#define N 10                                    // Розмір масиву для черги
// Функція створює двійковий файл цілих чисел.
void StvorenFile(FILE *p);
// Функція виводить на екран вміст двійкового файлу цілих чисел.
void DrucFile(FILE *p);
// Функція добавляє елемент у хвіст черги. Через  параметр mas передається масив (черга),
// n – розмір масиву, phead – вказівник на індекс вершини черги, ptail – вказівник на індекс
// хвоста черги, z – порядковий номер найбільшого елемента у файлі.
void AddCherga(int mas[], int n, int * phead, int * ptail, int z);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int i;
        int head = -1;                          // Індекс вершини черги
        int tail = -1;                          // Індекс хвоста черги
        int cherga[N];                          // Масив для черги
        char filename[120];                     // Змінна для імені файлу
        printf ("Введіть імя файлу\n");
        gets(filename);                         // Введення з клавіатури імені файлу
        FILE * fp;
        fp=fopen(filename, "wb");               // Відкриття двійкового файлу для запису
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        StvorenFile(fp);                        // Створення двійкового файлу
        fp=fopen(filename, "rb");               // Відкриття двійкового файлу для читання
        if (fp==NULL)
        {       puts("Файл не відкритий\n");
                exit(0);
        }
        printf ("Друк файлу\n");
        DrucFile(fp);                           // Виведення вмісту файлу на екран
```

```
        rewind(fp);                     // Встановлення поточного вказівника на початок файлу
        int x;
        int max;                                        // Змінна для найбільшого числа
        int index = -1;                                 // Змінна для номерів чисел у файлі
        fread(&x, sizeof(int), 1, fp);                  // Читаємо з файлу перше число
        max = x;                                        // Вважаємо його найбільшим
        index = 0;                                      // Його порядковий номер
        AddCherga(cherga, N, &head, &tail, index);      // Заносимо цей номер в чергу
        fread(&x, sizeof(int), 1, fp);                  // Читаємо з файлу друге число
        while (!feof(fp))                               // Поки не кінець файлу
        {       index++;                                // Збільшуємо номер числа
                if (x > max)            // Якщо число більше найбільшого поточного числа
                {       max = x;                        // Нове найбільше число
                        head = -1;                      // Очищення черги
                        tail = -1;
                        AddCherga(cherga, N, &head, &tail, index);      // Заносимо номер
                                                                        нового найбільшого числа в чергу
                }
                else
                        if (x == max)                   // Якщо число рівне найбільшому
                                // Заносимо його номер в чергу
                                AddCherga(cherga, N, &head, &tail, index);
                fread(&x, sizeof(int), 1, fp);          // Читаємо з файлу наступне число
        }
        fclose(fp);                                     // Закриваємо файл
        printf ("Найбільше число файлу = %d\n", max);
        printf ("Його індекси у файлі\n");
        // Виведення номерів найбільшого числа
        if (head < tail)
                for (i=head; i<=tail; i++)
                        printf ("%d  ", cherga[i]);
        else
        {       for (i=head; i<N; i++)
                        printf ("%d  ", cherga[i]);
                for (i=0; i<=tail; i++)
                        printf ("%d  ", cherga[i]);
        }
        printf ("\n");
        system("pause");
        return 0;
}
//-----------------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{       // Організовуємо цикл для введення цілих чисел з клавіатури і запису їх у файл,
        // що пов’язаний з вказівником p. Якщо введене число 999, то цикл припиняє
        // роботу. Це число не заноситься у файл.
        printf ("Введіть числа у файл. Останне число - 999\n");
        int x;
        scanf("%d", &x);        // Введення цілого числа з клавіатури
        while (x!=999)
        {       fwrite(&x, sizeof(int), 1, p);          // Записуємо число у файл
                scanf("%d", &x);                        // Вводимо чергове ціле число
```

```
        }
        fclose(p);                              // Закриття файлу
}
//-------------------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{       int x;
        fread(&x, sizeof(int), 1, p);           // Читаємо число x з файлу
        while (!feof(p))                        // Поки не кінець файлу
        {       printf("%d   ", x);             // Виводимо число x на екран
                fread(&x, sizeof(int), 1, p);   // Читаємо наступне число x з файлу
        }
        printf("\n");
}
//-------------------------------------------------------------------------------------------------------------------
void AddCherga(int mas[], int n, int * phead, int * ptail, int z)
{       if (*phead == -1)                       // Черга порожня
        {       *phead = 0;                     // Індекс вершини
                *ptail = 0;                     // Індекс хвоста
                mas[*ptail] = z;                // Записуємо номер числа в чергу
        }
        else
        {       (*ptail)++;                     // Збільшуємо індекс хвоста
                *ptail %= n;                    // Визначаємо його реальне значення
                if (*phead == *ptail)
                        printf("Черга повна. Добавити елемент не можна\n");
                else
                        mas[*ptail] = z;        // Записуємо номер числа в чергу
        }
}
```

```
Введіть імя файлу
D:\fff
Введіть числа у файл. Останне число - 999
2 -3 44 6 12 55 -11 43 32 -33 12 55
43 -43 5 5 6 55 34 2 3 55 55 0 -1
999
Друк файлу
2   -3   44   6   12   55   -11   43   32   -33   12   55   43   -43   5   5   6
   55   34   2   3   55   55   0   -1
Найбільше число файлу = 55
Його індекси у файлі
5  11  17  21  22
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Задачі

1. Створити файл структур із полями: назва виробу, тип виробу (дерев'яний, залізний, скляний і т.д.). За один перегляд файлу надрукувати спочатку назви дерев'яних виробів, потім залізних, далі всіх інших (зі збереженням початкового взаємного порядку в файлі у кожній з цих груп).
2. Дано два числові відрізки, які не перетинаються. З клавіатури вводиться послідовність із n дійсних чисел. Не зберігаючи послідовності в пам'яті, вивести спочатку числа, шо належать першому відрізку, потім числа другого відрізка, а далі вивести всі інші числа. Використати черги, які реалізовані у вигляді масиву.

☺ Індивідуальні завдання

Основний рівень

1. Створити текстовий файл. За один перегляд файлу роздрукувати його зміст, виводячи спочатку голосні букви, а потім інші (у порядку слідування у файлі).
2. Створити файл дійсних чисел. За один перегляд файлу вивести на екран спочатку суми кожних двох чисел, а потім різниці кожних двох. При виведенні зберігати порядок розташування чисел у файлі.
3. Створити текстовий файл. За один перегляд файлу вивести на екран кожне третє слово, а потім кожне четверте. При виведенні зберігати порядок розташування слів у файлі.
4. Створити файл дійсних чисел. За один перегляд файлу вивести спочатку всі від'ємні числа, а потім всі додатні. При виведенні зберігати порядок розташування чисел у файлі.
5. Створити файл цілих чисел. Переглядаючи його лише один раз, надрукувати спочатку всі двоцифрові числа, потім – всі інші. При виведенні зберігати порядок розташування чисел у файлі.
6. Створити файл цілих чисел. За один перегляд файлу вивести на екран спочатку всі парні числа, а потім всі непарні. При виведенні зберігати порядок розташування чисел у файлі.
7. Створити файл дійсних чисел. За один перегляд файлу підрахувати кількість чисел, що мають найбільшу абсолютну величину, та вивести за зростанням порядкові номери цих чисел у файлі.
8. Дано файл натуральних чисел. За один перегляд файлу, вивести спочатку числа, сума цифр яких більша за **a,** а потім решту. При виведенні зберігати порядок розташування чисел у файлі.
9. Створити файл дійсних чисел. За один перегляд без використання додаткових файлів надрукувати елементи файлу в такому порядку: спочатку всі числа, менші за **a**, потім – всі числа з відрізка **[a,b]**, а в кінці – всі інші числа, зберігаючи початковий взаємний порядок у кожній із цих груп чисел (**a** і **b** – задані числа, **a<b**).
10. Ввести послідовність натуральних чисел, у кінці якої 0. Не зберігаючи всієї послідовності в пам'яті, вивести порядкові номери найбільших чисел послідовності в порядку зростання їх номерів.
11. Ввести послідовність цілих чисел. Не зберігаючи всієї послідовності в пам'яті, вивести порядкові номери найменших чисел послідовності в порядку збільшення їх номерів .
12. Створити файл записів із полями: назва товару, ціна товару. За один перегляд файлу вивести назви всіх найдорожчих товарів у порядку слідування їх у файлі.
13. Дана послідовність цілих чисел. Не зберігаючи всієї послідовності в пам'яті, вивести всі числа (у порядку їх введення), сума цифр яких найменша.
14. Створити файл записів із полями: прізвище, номер телефону. За один перегляд файлу вивести прізвища абонентів, сума цифр номера телефону яких найменша (зберігаючи порядок їх розташування у файлі).
15. Створити текстовий файл. За один перегляд файлу створити новий файл, у якому будуть знаходитися всі найдовші слова - паліндроми, зберігаючи початковий порядок слів у файлі.

Підвищений рівень

1. Створити файл структур із полями: назва продукту, тип продукту (хлібний виріб, молочний, фрукти, овочі і т.д.). За один перегляд файлу надрукувати спочатку назви

хлібних виробів, потім молочних продуктів, далі всіх інших (зі збереженням початкового взаємного порядку в файлі у кожній з цих груп).

2. Створити файл структур із полями: прізвище студента, оцінки за сесію. За один перегляд файлу надрукувати спочатку прізвища всіх студентів, які склали сесію на відмінно, потім – без трійок, далі – без двійок, нарешті – мають заборгованість (зі збереженням початкового порядку в кожній з цих груп у файлі).
3. Створити файл структур із полями: назва міста, область, до якої належить місто (Кіровоградська, Київська, Львівська). За один перегляд файлу вивести спочатку міста Кіровоградської області, потім Київської, далі Львівської. При виведенні зберігати порядок розташування міст у файлі.
4. Дано непорожню послідовність слів, між словами пробіл, за останнім словом крапка. Надрукувати ці слова в такому порядку: всі слова, які складаються з одної букви, потім – двобуквені слова, нарешті – всі інші, зберігаючи початковий порядок слів.
5. Створити файл структур із полями: прізвище абітурієнта, його оцінки за вступні екзамени. За один перегляд файлу надрукувати спочатку прізвища абітурієнтів, які набрали більше 10 балів, потім – 9 або 10 балів (не пройшли за конкурсом), нарешті – отримали двійку (зі збереженням початкового порядку в файлі у кожній з цих груп).
6. Створити файл структур із полями: прізвище, номер телефону. За один перегляд файлу вивести спочатку інформацію про абонентів, номер телефону яких починається на 24, потім – на 55, нарешті – інші, зберігаючи початковий порядок запису абонентів у файл.
7. Створити файл структур із полями: прізвище студента, номер курсу. За один перегляд файлу надрукувати спочатку прізвища студентів першого курсу, потім – другого, і т.д., зберігаючи порядок введення записів у файл.
8. Створити файл структур із полями: прізвище абітурієнта, оцінки за вступні екзамени. За один перегляд файлу вивести інформацію про абітурієнтів, які набрали найбільшу кількість балів, потім – не отримали двійку і нарешті - всіх інших, зберігаючи порядок введення записів у файл.
9. Створити файл цілих чисел. За один перегляд файлу вивести спочатку порядкові номери найменших чисел, а потім – найбільших, зберігаючи порядок занесення чисел до файлу.
10. Створити текстовий файл. За один перегляд файлу створити два новий файли, у яких будуть знаходитися відповідно всі найдовші та найкоротші слова. Слова у нових файлах повинні зберігати порядок їх розташування у початковому файлі..
11. Створити файл цілих чисел. За один перегляд файлу створити новий файл, у якому будуть знаходитися спочатку всі від’ємні числа, потім – рівні нулю, нарешті – додатні числа , зберігаючи порядок розташування чисел у початковому файлі.
12. Створити файл структур із полями: назва іграшки, ціна, для якого віку створена іграшка. За один перегляд файлу вивести інформацію про іграшки спочатку для дітей до 5 років, потім – від 6 до 10 років, нарешті – для іншого віку, зберігаючи порядок розташування записів у файлі.
13. Створити файл структур із полями: прізвище, номер телефону. За один перегляд файлу вивести спочатку інформацію про абонентів АТС №2 (номер телефону починається з 2), потім – про абонентів АТС №3, нарешті – про всіх інших, зберігаючи порядок розташування записів у файлі.
14. Створити текстовий файл. За один перегляд файлу вивести спочатку слова довжиною не більше 5 символів, потім – усі слова, крім найдовших, нарешті – найдовші слова, зберігаючи початковий порядок слів у файлі.
15. Створити файл структур із полями: назва товару, ціна товару. За один перегляд файлу вивести спочатку інформацію про товари ціною до 100 грн., потім – ціною від 100 грн до 1000 грн., далі – дорожчі 1000 грн., зберігаючи порядок введення записів у файл.

Додаткові задачі

1. Чергою з пріоритетами називається послідовність елементів, кожному з яких приписано певний пріоритет (натуральне число). Над елементами такої черги можна виконувати операції добавлення елемента та вилучення елемента з найменшим пріоритетом. Створити діалогову програму, яка б через систему меню реалізовувала команди: ініціалізація черги (створення початкової черги), добавлення елемента, вилучення елемента, кінець роботи.
2. Дано три черги цілих чисел, упорядкованих за зростанням. Створити з них одну чергу з різних елементів теж упорядкованих за зростанням.
3. *Пошук у ширину*. Задано зв'язний неорієнтований граф із n вершин списками суміжностей. Пошук у ширину дозволяє відвідати всі вершини графа та отримати стовбурне дерево цього графа. Створити програму цього алгоритму. Також програма повинна виводити стовбурне дерево графа.
4. Використовуючи черги, написати програму додавання та множення дуже великих цілих чисел.

Питання для самоконтролю

1. Що таке динамічна пам'ять? Для чого вона служить?
2. Що таке вказівник?
3. Основні операції над чергами.
4. Який механізм роботи черги?
5. Куди необхідно добавляти елементи до черги та звідки вилучати?
6. Що може трапитись при рекурсивному добавлені елементів до черги?
7. Де зберігаються елементи динамічних структур?
8. У яких випадках доцільно використовувати масив або чергу?

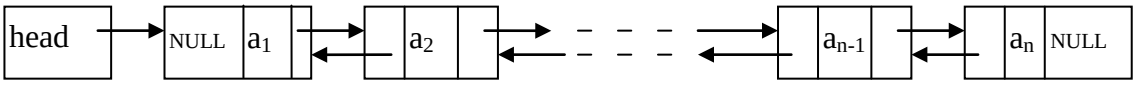
Практичне заняття № 18

*Тема: **Двозв'язні лінійні списки***

Студент повинен знати: поняття двозв'язного лінійного списку та синтаксис оголошення типу його елементів, створення та ініціалізацію списку, операції добавлення та вилучення елементів.

Теоретичні відомості

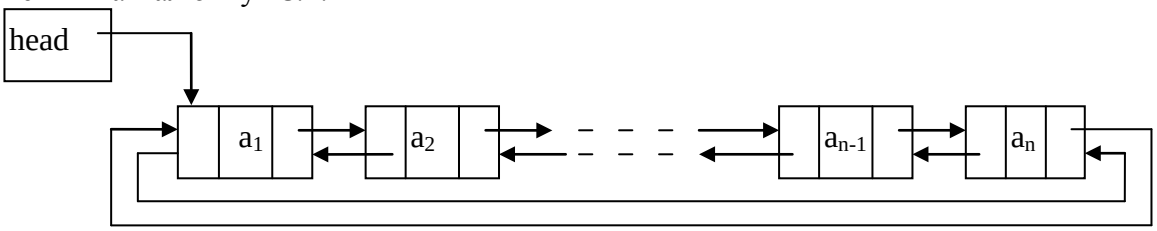
Двозв'язний лінійний список – це лінійний список, у якому попередній елемент показує на наступний, а наступний – на попередній. Такий список зображений на малюнку 18.1.



Мал. 18.1. Двозв'язний лінійний список.

Вказівник head показує на початок списку, хоч поняття «початок» та «кінець» для такого списку рівноправні. На малюнку head зображений зліва списку, але його можна було зобразити справа. У двозв'язному списку можна переглядати його елементи в обох напрямках.

Двозв'язний циклічний список (кільце) – це двозв'язний лінійний список, у якому останній елемент показує на перший, а перший елемент – на останній. Такий список зображений на малюнку 18.2.



Мал. 18.2. Двозв'язний циклічний список.

Елемент двозв'язного списку є структурою із полем (полями) для даних і двома полями для вказівників: next (наступний) та prev (попередній). Оголошення типу цього елемента здійснюється за синтаксисом:

```
typedef struct elem
 {      <тип даних> dani;
        struct elem * next;
        struct elem * prev;
 }      element;
typedef element * ptr;
```

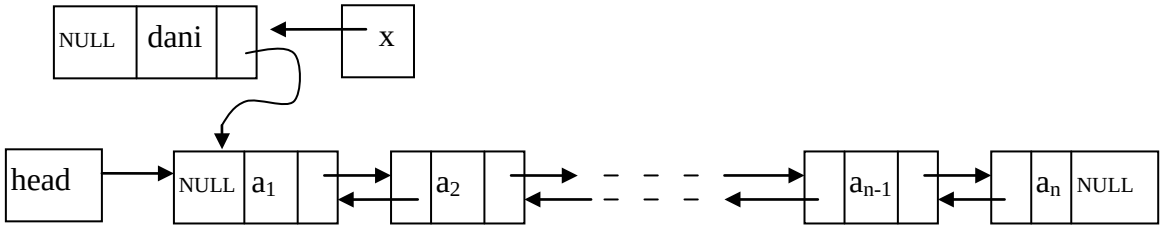
Після визначення типів можна оголошувати вказівники. Наприклад,

ptr head, current;

Вказівник head потрібний для створення та роботи зі списком, а current – вказівник на поточний елемент. Спочатку доцільно створити один елемент списку і присвоїти його адресу вказівнику head, хоч можливі інші варіанти початкової ініціалізації head. Потім створюють решту списку, виконуючи операцію добавлення елементів до списку необхідну кількість разів.

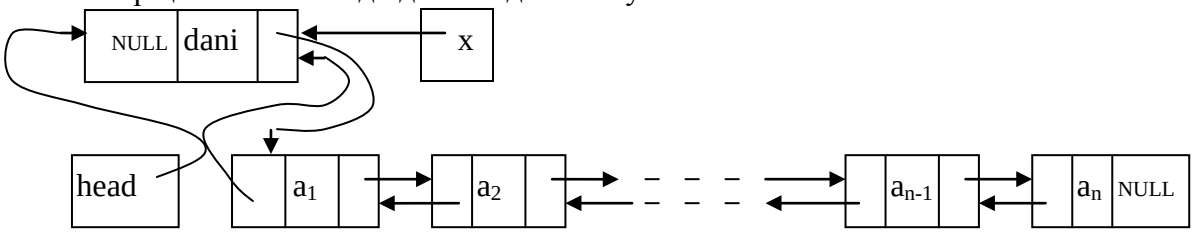
Операція добавлення нового елемента до списку полягає в тому, що спочатку в динамічній пам'яті створюється новий елемент, а потім він під'єднується до списку. Розглянемо різні варіанти цієї операції.

На наступному малюнку 18.3 зображено створення нового елемента X, який буде добавлятися до вершини списку.



Мал. 18.3. Створюємо елемент, на який вказує вказівник x.

Тепер цей елемент під'єднаний до списку.



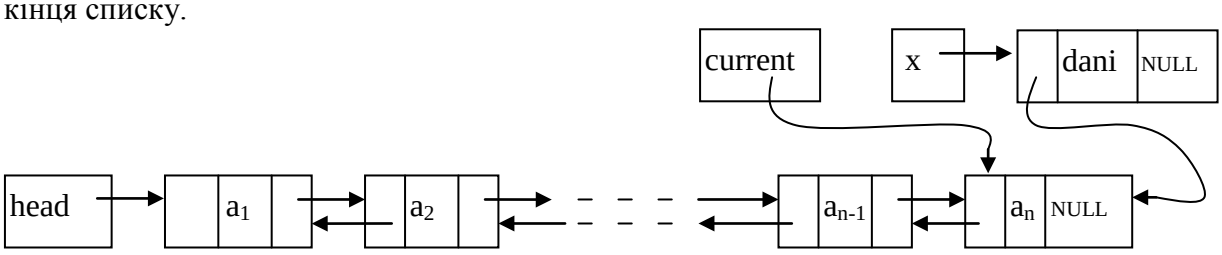
Мал. 18.4. Під'єднання елемента до списку.

Операцію добавлення елемента до вершини списку можна реалізувати наступною функцією.

ptr AddElemV(ptr head, <тип_даних> z)

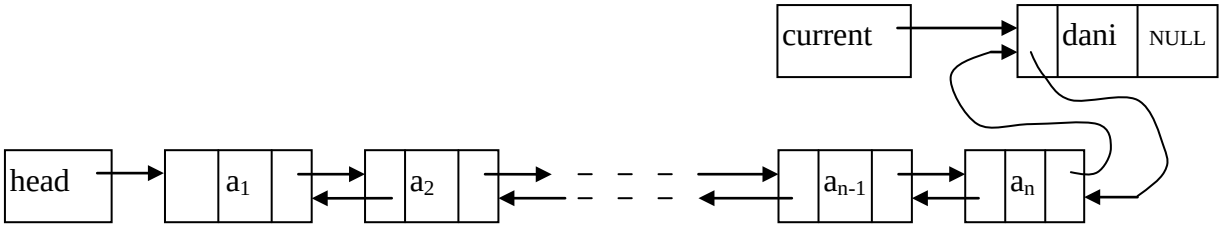
```
{       ptr x;
        x = (ptr) malloc (sizeof(element));   // Виділення динамічної пам'яті для елемента
        x->dani = z;                          // Занесення даних
        x->prev = NULL;                       // Визначення вказівника prev
        x->next = head;                       // Визначення вказівника next
        head->prev = x;                       // Під'єднуємо елемент до списку
        head = x;                             // Зміна head
        return head;                          // Повертаємо head
}
```

Зобразимо на малюнку 18. 5 створення нового елемента, що буде добавлятися до кінця списку.



Мал. 18.5. Створюємо елемент, на який вказує вказівник x.

Після добавлення елемента список матиме вигляд.



Мал. 18.6. Під'єднання елемента до списку.

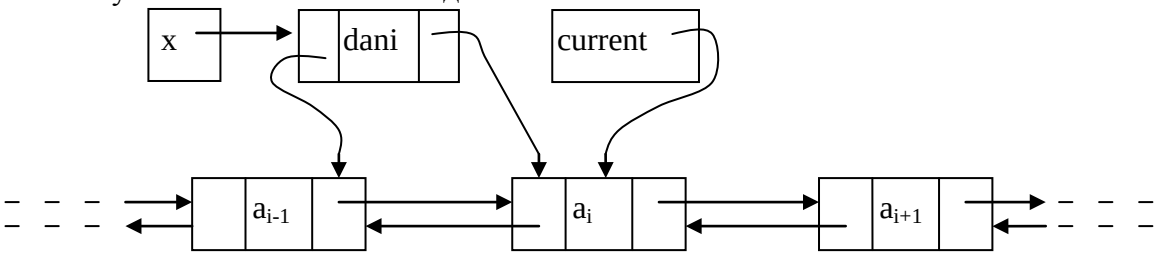
Відповідна функція має вигляд, якщо вважати current вказівником на останній елемент списку.

ptr AddElemK(ptr current, <тип_даних> z)

```
{       ptr x;
        x = (ptr) malloc (sizeof(element));   // Виділення динамічної пам'яті для елемента
        x->dani = z;                          // Занесення даних
        x->prev = current;                    // Визначення вказівника prev
        x->next = NULL;                       // Визначення вказівника next
```

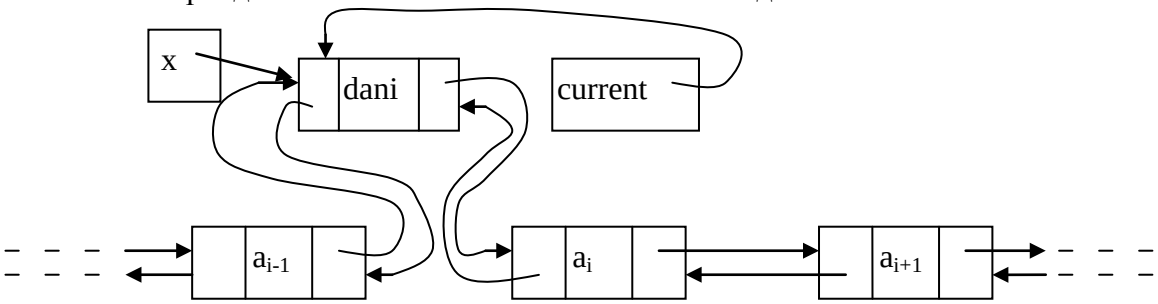
```
        current->next = x;              // Під'єднуємо елемент до списку
        current = x;                    // Зміна current
        return current;                 // Повертаємо current
}
```

Тепер добавлятимемо новий елемент між двома наявними елементами списку. Нехай відомий вказівник current на деякий внутрішній елемент списку. Добавимо спочатку новий елемент зліва від current.



Мал. 18.7. Створюємо елемент, на який вказує вказівник x.

Після приєднання елемента список матиме вигляд.



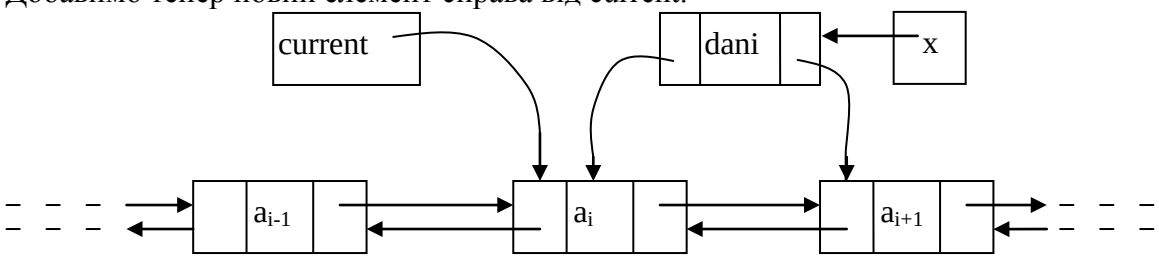
Мал. 18.8. Під'єднання елемента до списку.

Функція добавлення має вигляд, якщо вважати current вказівником на і-й елемент списку.

ptr AddElemSL(ptr current, <тип_даних> z)

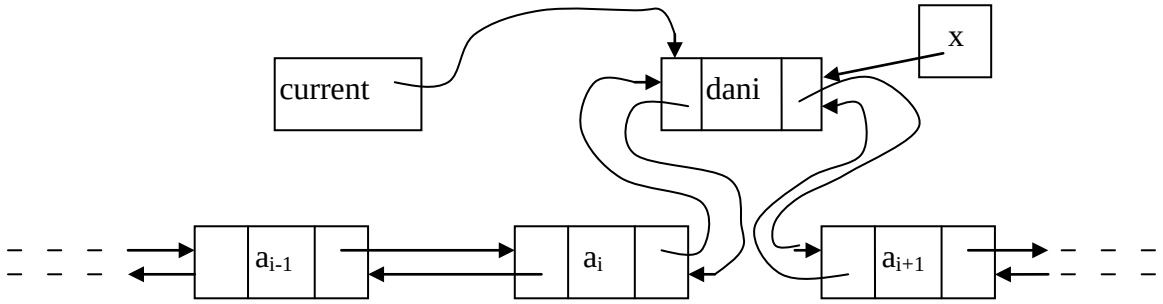
```
{       ptr x;
        x = (ptr) malloc (sizeof(element));   // Виділення динамічної пам'яті для елемента
        x->dani = z;                          // Занесення даних
        x->next = current;                    // Визначення вказівника next
        x->prev = current->prev;              // Визначення вказівника prev
        current->prev->next = x;              // Під'єднуємо елемент зліва до списку
        current->prev = x;                    // Під'єднуємо елемент справа до списку
        current = x;                          // Змінюємо current
        return current;                       // Повертаємо current
}
```

Добавимо тепер новий елемент справа від current.



Мал. 18.9. Створюємо елемент, на який вказує вказівник x.

Після приєднання елемента список матиме вигляд.



Мал. 18.10. Під'єднання елемента до списку.

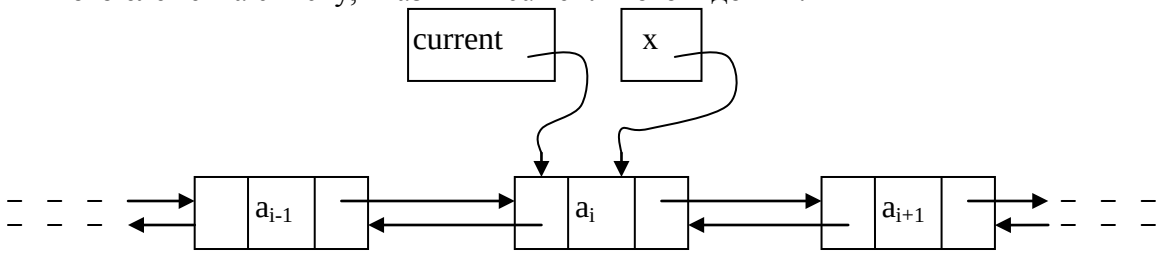
Функція добавлення має вигляд, якщо вважати current вказівником на і-й елемент списку.

ptr AddElemSR(ptr current, <тип_даних> z)

```
{       ptr x;
        x = (ptr) malloc (sizeof(element));   // Виділення динамічної пам'яті для елемента
        x->dani = z;                          // Занесення даних
        x->next = current->ntxt;              // Визначення вказівника next
        x->prev = current;                    // Визначення вказівника prev
        current->next->prev = x;              // Під'єднуємо елемент справа до списку
        current->next = x;                    // Під'єднуємо елемент зліва до списку
        current = x;                          // Змінюємо current
        return current;                       // Повертаємо current
}
```

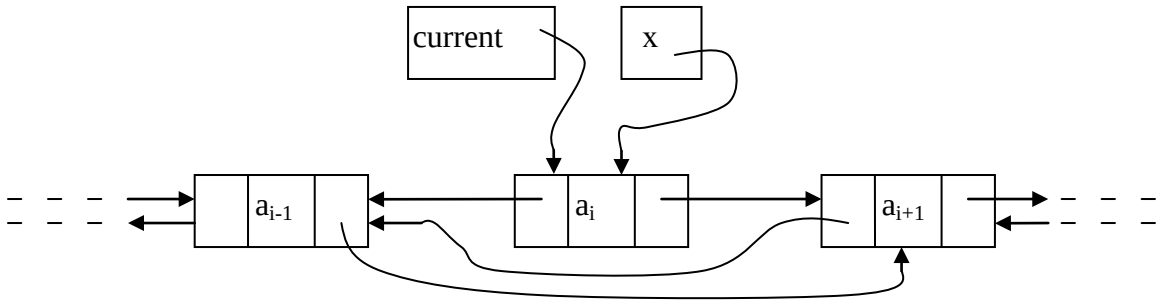
Ми розглянули можливі варіанти добавлення елемента до двозв'язного лінійного списку. Вони аналогічні для кільця (можливо з невеличкими уточненнями).

Програмна реалізація операції вилучення теж можлива з різними варіаціями, залежно від того, який елемент списку вилучається. Опишемо лише операцію вилучення внутрішнього елемента списку, вказівник current якого відомий.



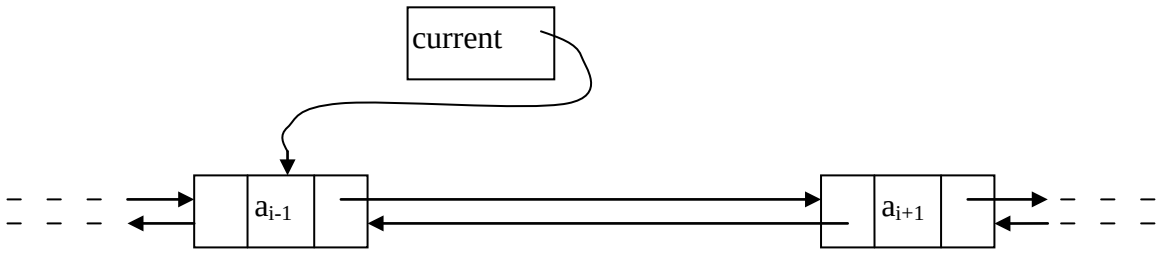
Мал. 18.11. Вказівник x вказує на елемент, який буде вилучатися.

На малюнку 18.11 зображені три елементи списку – (i-1)-й, і-й, (i+1)-й. Припустимо, що ми знаємо і-й елемент, і нехай його вказівник current. Треба вилучити і-й елемент. Для цього спочатку треба визначити вказівник x на цей елемент. Це робиться через присвоювання x=current.



Мал. 18.12. Обхід елемента.

Наступним кроком операції вилучення є зміна посилань (i-1)-го та (i+1)-го елементів так, як показано на малюнку 18.12. Це робиться через послідовність присвоювань current->prev->next=current->next, current->next->prev=current>prev. Тепер можна змінити значення вказівника current та вилучати і-й елемент. Отримаємо малюнок 18.13.



Мал. 18.13. Список без вилученого елемента.

Операцію вилучення можна реалізувати наступною функцією.

```
ptr DelElem(ptr current)
{       ptr x;
        x = current;                            // Запам'ятовуємо елемент, який буде вилучатися
        current->prev->next = current->next;    // Обхід елемента по вказівнику next
        current->next->prev = current->prev;    // Обхід елемента по вказівнику prev
        current = current->prev;                // Змінюємо current
        free(x);
        return current;                         // Повертаємо current
}
```

Якщо необхідно переглядати елементи списку в обох напрямках, доцільно такий список реалізувати у вигляді кільця. Початкову ініціалізацію кільця можна здійснити так:

```
head = (ptr) malloc (sizeof(element));
head->dani = <якесь початкове значення>;
head->next = head;
head->prev= head;
```

Перегляд елементів списку зазвичай організовується з використанням циклу while.

Приклад 1

Дано текстовий файл, що містить слова, розділені пробілами. За один перегляд файлу для кожного слова підрахувати, скільки разів воно зустрічається в тексті. Вивести слова за спаданням частоти повторень, а слова з однаковою частотою упорядкувати за алфавітом.

Аналіз задачі

Спочатку створимо текстовий файл, використавши для цього функцію. Інформацію у файл записуємо рядками. Процес створення файлу триватиме до тих пір, поки не буде введений порожній рядок. Цей рядок не запишеться у файл, а всі попередні рядки будуть складатися із слів, що розділяються хоча б одним пробілом.

Тепер будемо послідовно переглядати файл від початку до кінця. Файл читаємо рядками, а рядки розбиваємо на слова. Отримані слова заносимо у кільце. Для кожного слова враховуємо кількість повторень. Після перегляду файлу отримаємо кільце слів. Далі застосуємо алгоритм вставок для упорядкування кільця згідно з умовою задачі.

```
#include "stdafx.h"
#include "windows.h"
#include "string.h"
// Визначення типу element для елементів двозв'язного списку
typedef struct elem
        {       char slovo[20];         // Поле для слова
                int kilk;               // Кількість слів
                struct elem * next;     // Вказівник next
                struct elem * prev;     // Вказівник prev
        }       element;
// Визначення типу ptr для вказівників на елементи списку
```

```
typedef element * ptr;
// Функція створює текстовий файл
void StvorenFile(FILE *p);
// Функція виводить текстовий файл на екран
void DrucFile(FILE *p);
// Функція добавляє елемент до двозв'язного списку справа від елемента, на який вказує
// current. Через вказівник z у функцію передається слово. Функція повертає вказівник на
// добавлений елемент
ptr AddElemSR(ptr current, char * z);
// Функція перевіряє чи міститься слово у двозв'язному списку. Якщо слово є в списку, то
// збільшуємо його кількість, інакше - добавляємо його у список. Двозв'язний список
// створюється у вигляді кільця. Через параметр head у функцію передається вказівник на
// кільце і він повертається функцією. Параметр z служить для передачі слова.
ptr PoiskVstavky(ptr head, char * z);
// Функція виводить на екран слова зі списку. Через параметр head передається вказівник
// на кільце.
void DrukSpisok(ptr head);
// Функція упорядковує список слів за спаданням кількості повторень, а при одинаковій
// кількості – за алфавітом. Через параметр head передається вказівник на кільце, який
// також повертається функцією.
ptr SortSpisok(ptr head);
// Функція вилучає елемент зі списку.
ptr DelElem(ptr current);

int _tmain(int argc, _TCHAR* argv[])
{       int k;                          // Довжина рядка
        char filename[120];             // Змінна для імені файлу
        printf ("Vvedit imja faylu\n");
        gets(filename);                 // Введення імені файлу з клавіатури
        FILE * fp;                      // Вказівник на потік
        fp=fopen(filename, "w");        // Відкриття текстового файлу для запису
        if (fp==NULL)
        {       puts("file ne vidkryvsja");
                exit(0);
        }
        StvorenFile(fp);                // Створення текстового файлу
        fp=fopen(filename, "r");        // Відкриття текстового файлу для читання
        if (fp==NULL)
        {       puts("file ne vidkryvsja");
                exit(0);
        }
        printf ("Druk file\n");
        DrucFile(fp);                   // Виведення файлу на екран
        rewind(fp);                     // Встановлення поточного вказівника на початок файлу
        ptr head, cur;                  // Оголошення вказівників для роботи з кільцем
        // Створення кільці з одним фіктивним елементом
        head = (ptr) malloc(sizeof(element));       // Виділення динамічної пам'яті
        strcpy(head->slovo, "");        // Заносимо порожнє слово
        head->kilk = 0;                 // Кількість слів
        head->next = head;              // Створюємо кільце через значення вказівників
        head->prev = head;
        char rydok[128];                // Змінна для рядка
```

```
	char s[2]=" ";				// Рядок розділяючих символів між словами
	char *s1;
	fgets(rydok, 127, fp);			// Читаємо рядок з файлу
	while (!feof(fp))			// Поки не кінець файлу
	{	k = strlen(rydok);		// Визначаємо довжину рядка
		rydok[k-1] = '\0';		// Замінюємо символ ‘\n’ на ‘\0’
		s1=strtok(rydok, s);		// s1 – перше слово рядка
		while (s1!=NULL)
		{	head = PoiskVstavky(head, s1);	// Заносимо s1 в кільце
			s1=strtok(NULL, s);		// s1 – наступне слово рядка
		}
		fgets(rydok, 127, fp);		// Читаємо наступний рядок з файлу
	}
	fclose(fp);				// Закриваємо файл
	puts("Vyvod sliv zi spysku");
	DrukSpisok(head);			// Виведення слів зі списку
	printf ("\n");
	head = SortSpisok(head);		// Упорядкування списку слів
	puts("Vidsortovany spisok sliv");
	cur = head->next;
	while (cur != head)    // Виведення упорядкованого списку слів і знищення кільця
	{	printf("%s (%d)   ", cur->slovo, cur->kilk);
		cur = DelElem(cur);
	}
	printf ("\n");
	system("pause");
	return 0;
}
//-----------------------------------------------------------------------------------------------------------------
void StvorenFile(FILE *p)
{	// Створення текстового файлу із рядків слів
	printf ("Vvedit rjadky u file. Osnanniy rjadok - enter\n");
	char s[128];
	fgets(s, 127, stdin);			// Вводимо рядок з клавіатури
	while (strcmp(s, "\n")!=0)		// Поки не введений порожній рядок
	{	fputs(s, p);			// Записаємо рядок у файл
		fgets(s, 127, stdin);		// Вводимо черговий рядок з клавіатури
	}
	fclose(p);				// Закриваємо файл
}
//-----------------------------------------------------------------------------------------------------------------
void DrucFile(FILE *p)
{	// Виведення файлу на екран
	char s[128];
	fgets(s, 127, p);			// Читаємо рядок з файлу
	while (!feof(p))			// Поки не кінець файлу
	{	fputs(s, stdout);		// Виводимо рядок на екран
		fgets(s, 127, p);		// Читаємо наступний рядок з файлу
	}
}
//-----------------------------------------------------------------------------------------------------------------
ptr AddElemSR(ptr current, char * z)
```

```
{       // Добавлення елемента у двозв'язний список (кільце)
        ptr x;
        x = (ptr) malloc (sizeof(element));   // Виділення динамічної пам'яті для елемента
        strcpy(x->slovo, z);;                 // Заносимо слово
        x->kilk = 1;                          // Кількість слів
        x->next = current->next;              // Визначення вказівника next
        x->prev = current;                    // Визначення вказівника prev
        current->next->prev = x;              // Під'єднуємо елемент справа до списку
        current->next = x;                    // Під'єднуємо елемент зліва до списку
        current = x;                          // Змінюємо current
        return current;                       // Повертаємо current
}
//-------------------------------------------------------------------------------------------------------------
ptr PoiskVstavky(ptr head, char * z)
{       // Пошук у кільці слова, яке передається через параметр z. Якщо слова немає, то
        // воно добавляється у кільце справа від head (у напрямку вказівника next). Якщо ж
        // слово є в кільці, то збільшується його кількість. Функція повертає head.
        ptr x;
        x = head->next;                       // Вказівник на перше слово списку
        while ((x!=head)&&(strcmp(x->slovo, z)!=0))       // Рух по кільцю – пошук слова z
                x = x->next;
        if (strcmp(x->slovo, z)==0)           // Якщо слово z є в кільці
                x->kilk++;                    // Збільшуємо його кількість
        else                                  // Інакше, заносимо слово в список
                x = AddElemSR(x, z);
        return head;                          // Повертаємо head
}
//-------------------------------------------------------------------------------------------------------------
void DrukSpisok(ptr head)
{       // Виведення спику слів на екран
        ptr x;
        x = head->next;                       // Вказівник на перше слово списку
        while (x!=head)                       // Поки не пройшли весь список
        {       printf("%s (%d)   ", x->slovo, x->kilk);       // Виводимо слово і його частоту
                x = x->next;                  // Перехід на нступний елемент списку
        }
        printf("\n");
}
//-------------------------------------------------------------------------------------------------------------
ptr SortSpisok(ptr head)
{       // Сортування двозв'язного списку (кіьця) методом вставок за спадання кількості
        // повторень слів, а слова однакової кількості сортуються за алфавітом. Параметр
        // head відповідає за весь список. Після завершення сортування він повертається
        // функцією.
        ptr x,y, current;             // Вказівники на елементи списку
        x = head->next->next;         // x – вказівник на другий елемент списку
        // Вважаємо, що початок списку з одного першого елемента вже відсортований
        while (x!=head)               // Поки не завершений обхід списку
        {       y=x->prev;            // y – вказівник на останній елемент відсортованої
                                                                частини списку
    // Пошук місця у відсортованій частині списку для вставки елемента, на який вказує x.
                // Після завершення циклу вказівник y визначає місце встаки
```

```
            while ((x->kilk > y->kilk)&&(y!=head))
                  y = y->prev;
            // Подальше уточнення місця встаки для слів з однаковою кількістю
            // повторень
            if (x->kilk == y->kilk)
            while ((x->kilk == y->kilk)&&(strcmp(x->slovo, y->slovo) < 0)&&(y!=head))
                  y = y->prev;
            // Після завершення цього циклу місце вставки визначає вказівник y
            // current – вказівник на елемент, який спочатку треба вилучити зі списку,
            // а потім добавити у список у визначене місце вставки
            current = x;
            x = x->next;              // Змінюємо x для руху по списку.
            // x – вказівник на наступний елемент, для якого буде виконуватися вставка
            // при наступній ітерації циклу
            // Обхід елемента, на який вказує current (вилучення елемента зі списку)
            current->prev->next = x;
            x->prev = current->prev;
            // Вставка елемента, на який вказує current, у список
            current->prev = y;
            current->next = y->next;
            y->next->prev = current;
            y->next = current;
      }
      return head;
}
//-------------------------------------------------------------------------------------------------------------
ptr DelElem(ptr current)
{     // Вилучення елемента зі списку, на який вказує параметр current
      ptr x;
      x = current;                        // Запам’ятовуємо елемент, який буде вилучатися
      current->prev->next = current->next;      // Обхід елемента по вказівнику next
      current->next->prev = current->prev;      // Обхід елемента по вказівнику prev
      current = current->next;                  // Змінюємо current
      free(x);                                  // Звільнення динамічної пам’яті
      return current;                           // Повертаємо current
}
```

```
Vvedit imja faylu
D:\fff
Vvedit rjadky u file. Osnanniy rjadok - enter
пилип прилип до стола
бобер сова сом сова кабан
сова до до до слон пилип прилип пилип прилип
вікно рама мама тато мама абрикос яблуко

Druk file
пилип прилип до стола
бобер сова сом сова кабан
сова до до до слон пилип прилип пилип прилип
вікно рама мама тато мама абрикос яблуко
Vyvod sliv zi spysku
яблуко (1)   абрикос (1)   тато (1)   мама (2)   рама (1)   вікно (1)   слон (1)
   кабан (1)   сом (1)   сова (3)   бобер (1)   стола (1)   до (4)   прилип (3)
  пилип (3)

Vidsortovany spisok sliv
до (4)   пилип (3)   прилип (3)   сова (3)   мама (2)   абрикос (1)   бобер (1)
  вікно (1)   кабан (1)   рама (1)   слон (1)   сом (1)   стола (1)   тато (1)
 яблуко (1)
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Задачі

1. Розв’язати задачу з прикладу 1, реалізувавши двозв’язний список у вигляді масиву.
2. У рядку записаний многочлен від змінної x довільного степеня з цілими коефіцієнтами. Його одночлени можуть бути не упорядковані за степенями x, причому одночлени одного степеня можуть повторюватися. Наприклад,

   8x↑2-4x↑5+12-14x↑4+6x↑2-x↑4.

   Звести подібні члени многочлена та надрукувати його за спаданням степенів.

☺ Індивідуальні завдання

Основний рівень

1. Із клавіатури вводяться слова. Зберігаючи порядок введення, вивести спочатку однобуквені слова, потім двобуквені і т. д.
2. Із клавіатури вводиться послідовність натуральних чисел, що закінчується нулем. Упорядкувати цю послідовність за зростанням і вивести результат на екран, вказавши кількість повторень кожного числа у послідовності.
3. Із клавіатури вводиться послідовність натуральних чисел, що закінчується нулем. Упорядкувати цю послідовність за спаданням і вивести результат на екран, вказавши кількість повторень кожного числа у послідовності.
4. Із клавіатури вводяться слова. Упорядкувати ці слова за алфавітом, вказавши кількість повторень кожного слова.
5. Із клавіатури вводиться послідовність із n цілих чисел. Зберігаючи порядок введення, вивести спочатку одноцифрові числа, потім двоцифрові і т.д.
6. Дано текстовий файл. Підрахувати, скільки разів зустрічається у файлі кожна велика літера латинського алфавіту, і вивести результат за алфавітом.
7. Дано текстовий файл. Підрахувати, скільки українських слів зустрічається у файлі, що починаються з приголосної. Слова у файлі розділяються одним пробілом. Вивести знайдені слова за алфавітом, вказавши кількість повторень кожного слова.
8. Дано текстовий файл. Підрахувати, скільки українських слів зустрічається у файлі, що починаються з голосної. Слова у файлі розділяються одним пробілом. Вивести знайдені слова за алфавітом, вказавши кількість повторень кожного слова.
9. Дано текстовий файл. Підрахувати, скільки разів зустрічається у файлі кожна мала буква українського алфавіту, і вивести результат за алфавітом.
10. Із клавіатури вводиться послідовність цілих чисел, що не містить нулів, але закінчується нулем. Зберігаючи порядок введення, вивести спочатку одноцифрові від’ємні числа, потім двоцифрові від’ємні і т.д. За від’ємними числами в аналогічному порядку потім вивести додатні числа.
11. Із клавіатури вводиться послідовність із n дійсних чисел. Упорядкувати цю послідовність за спаданням і вивести результат на екран, вказавши кількість повторень кожного числа у послідовності.
12. Із клавіатури вводиться послідовність натуральних чисел, що закінчується нулем. Упорядкувати цю послідовність за зростанням сум цифр чисел і вивести результат на екран, вказавши кількість повторень кожного числа у послідовності.
13. Із клавіатури вводяться слова. Упорядкувати слова, довжина яких від 3 до 10 символів, за алфавітом, вказавши кількість повторень кожного слова.
14. Дано текстовий файл. Підрахувати, скільки українських слів зустрічається у файлі, що починаються та закінчуються приголосною. Слова у файлі розділяються одним пробілом. Вивести знайдені слова за алфавітом, вказавши кількість повторень кожного слова.
15. Дано текстовий файл. Підрахувати, скільки українських слів зустрічається у файлі, що починаються та закінчуються голосною. Слова у файлі розділяються одним пробілом. Вивести знайдені слова за алфавітом, вказавши кількість повторень кожного слова.

Підвищений рівень

1. Дано текст. Серед літер цього тексту особливу роль відіграє знак *, поява якого в тексті означає відміну попередньої літери тексту (якщо знаків декілька, то знищується декілька літер). Надрукувати даний текст з урахуванням ролі знаку *.
2. Дано текстовий файл, який містить латинські літери та цифри (1..9). Цифра означає кількість кроків, на які треба повернутися при перегляді файлу, і вилучити відповідний символ. Вивести перетворений текст.
3. Дано текстовий файл, який містить латинські літери та спеціальні маркери форматування. Маркер форматування складається із символу #, за яким записано букву і дві цифри (наприклад, #R31). Букви позначають: R – заміна на пробіл, D – вилучення символу, A – добавлення символу $. Перша цифра означає кількість слів, на які треба повернутися при перегляді файлу, а друга – номер символу в слові. Якщо слова чи символу з потрібним номером немає, то маркер пропускається. Вивести перетворений текст.
4. Дано текстовий файл, що містить інформацію про користувачів телефоном. У кожному рядку файлу записано прізвище, номер телефону та заборгованість, що розділені пробілами. Інформація у файл заносилася тривалий час, причому попередня інформація не коригувалася, а нові дані дописувалися у файл. Тому в файлі може зустрічатися інформація про одних і тих же користувачів, але з різними даними про заборгованість. Вивести інформацію про користувачів телефоном за спаданням заборгованості з урахуванням усіх даних. Користувачів із однаковою заборгованістю вивести за алфавітом.
5. Дано текстовий файл, що містить інформацію про користувачів водогоном. У кожному рядку файлу записано прізвище, адреса та заборгованість, що розділені пробілами. Інформація у файл заносилася тривалий час, причому попередня інформація не коригувалася, а нові дані дописувалися у файл. Тому в файлі може зустрічатися інформація про одних і тих же користувачів, але з різними даними про заборгованість. Вивести інформацію про користувачів водогоном за спаданням заборгованості з урахуванням всіх даних. Користувачів із однаковою заборгованістю вивести за алфавітом.
6. Дано текст, що закінчується знаком %. Серед слів цього тексту особливу роль відіграє знак *, поява якого в тексті означає відміну попереднього слова (якщо знаків декілька підряд, то знищуються декілька слів). Надрукувати даний текст з урахуванням ролі знаку *.
7. У автопарку кожного дня у текстовий файл дописувалася інформація. Кожний рядок файлу містить номер автобуса, прізвище водія, кількість відпрацьованих годин, які розділені пробілами. Дані у файл заносилися протягом місяці. Вивести інформацію про водіїв за спаданням кількості відпрацьованих годин, а при рівній кількості годин - вивести за алфавітом прізвищ.
8. Дано текстовий файл, що містить інформацію про іграшки. У кожному рядку файлу записана назва іграшки та ціна, що розділені пробілами. Інформація у файл заносилася тривалий час, причому попередня інформація не коригувалася, а нові дані дописувалися до кінця файлу. Тому в файлі може зустрічатися інформація про одні і ті ж іграшки, але з різною ціною. Вивести остаточну інформацію про іграшки за зростанням ціни, а іграшки з однаковою ціною вивести за алфавітом.
9. Дано текстовий файл, що містить інформацію про іграшки. У кожному рядку файлу записана назва іграшки та ціна, що розділені пробілами. Інформація у файл заносилася тривалий час, причому попередня інформація не коригувалася, а нові дані дописувалися до кінця файлу. Тому в файлі може зустрічатися інформація про одні і ті ж іграшки, але з різною ціною. Вивести остаточну інформацію про іграшки за зростанням ціни, а іграшки з однаковою ціною вивести за алфавітом.

10. На складі кожного дня у текстовий файл дописувалася інформація про надходження товарів. Кожний рядок файлу містить назву товару, кількість штук, які розділені пробілами. Дані у файл заносилися протягом місяці. Вивести інформацію про надходження товарів за місяць за зростанням кількості, а при рівній кількості - вивести назви товарів за алфавітом.
11. Дано текстовий файл. За один перегляд файлу вивести його зміст у такому порядку: спочатку всі українські слова за алфавітом, що починаються з голосної, та кількість їх входжень у файл, а потім аналогічні дані про слова, що починаються з приголосної.
12. Дано текстовий файл. За один перегляд файлу вивести його зміст у такому порядку: спочатку всі українські слова, що починаються і закінчуються голосною, та кількість їх входжень у файл, а потім аналогічні дані про слова, що починаються і закінчуються приголосною.
13. Дано текст, що містить слова, розділені пробілами. Перетворити його таким чином: з кожного наступного слова вилучити всі літери, на які починалися два попередні слова (з першого – нічого не вилучати, а з другого – вилучити літеру, на яку починалося перше слово). Результат перетворення тексту вивести на екран.
14. Дано файл дійсних чисел. Вивести спочатку всі числа за зростанням, а потім за спаданням, вказавши кількість повторень чисел у файлі. Вивести також кількість мінімальних та максимальних чисел.
15. Дано файл цілих чисел. Підрахувати кількість повторень кожного числа у файлі. Вивести інформацію про числа за зростанням кількості цифр, а при однаковій кількості цифр - вивести за зростанням величини.

Питання для самоконтролю

1. За яким синтаксисом оголошується тип елемента двозв’язного списку?
2. Дайте означення двозв’язного лінійного списку.
3. Дайте означення двозв’язного циклічного списку.
4. Які можливі варіанти добавлення елемента до двозв’язного лінійного списку? Чим вони відрізняються?
5. Які можливі варіанти добавлення елемента до двозв’язного циклічного списку? Чим вони відрізняються?
6. Розгляньте можливі варіанти вилучення елемента із двозв’язного лінійного списку. Напишіть відповідні функції.
7. Розгляньте можливі варіанти вилучення елемента із двозв’язного циклічного списку. Напишіть відповідні функції.
8. Як можна реалізувати двозв’язний список у вигляді масиву? Які переваги та недоліки такої реалізації?
9. Напишіть різні варіанти початкової ініціалізації двозв’язного списку.
10. Коли доцільно використовувати однозв’язний список, а коли двозв’язний?

Практичне заняття № 19

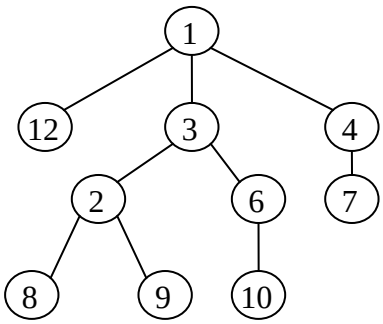
Тема: ***Двійкові дерева***

Студент повинен знати: означення дерева та бінарного дерева, синтаксис оголошення типу елементів бінарного дерева, алгоритми обходу дерев, операції пошуку, добавлення та вилучення елементів.

Теоретичні відомості

Дерево – це сукупність елементів, що називаються *вузлами* (один із яких визначається як *корінь*), і відношень («батьківських»), які утворюють ієрархічну структуру вузлів. Формально *дерево* можна рекурентно визначити наступним чином.

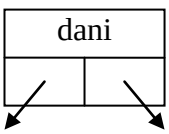
1. Один вузол є деревом. Цей же вузол є коренем цього дерева.
2. Нехай n - це вузол, а T_1, T_2, …, T_k – дерева з коренями n_1, n_2, …, n_k відповідно. Можна побудувати нове дерево, зробивши n батьком вузлів n_1, n_2, …, n_k. У цьому дереві n буде коренем, а T_1, T_2, …, T_k – піддеревами цього кореня. Вузли n_1, n_2, …, n_k називаються *синами* вузла n.



Мал. 19.1

Будемо вважати, що сини вузла впорядковані зліва направо, тобто розглядаємо *впорядковані орієнтовані дерева*. На малюнку 19.1 зображений приклад такого дерева. Коренем у ньому є вузол 1. *Шляхом із* вузла n_1 до вузла n_k називається послідовність вузлів n_1, n_2, …, n_k, де для всіх i, $1 \le i \le k-1$, вузол n_i є батьком вузла n_{i+1}. *Довжиною шляху* називається число, яке менше на 1 від числа вузлів, що утворюють цей шлях. Прикладом шляху довжини 2 є шлях від вузла 3 до вузла 9 для зображеного дерева. Якщо існує шлях від вузла *a* до *b*, то *a* називається *предком* вузла *b*, а вузол *b* – *нащадком* вузла *a*. Корінь немає предка. Вузол, що немає нащадків, називається *листком*. *Висотою вузла* називається довжина найдовшого шляху від цього вузла до якогось листка. *Висота дерева* – це висота кореня. *Глибиною* (*рівнем*) вузла називається довжина шляху від кореня до цього вузла. Корінь має рівень 0, а його сини – рівень 1, і т.д.

Двійкове (або *бінарне*) дерево – це орієнтоване дерево, але воно не є упорядкованим. Двійкове дерево може бути або порожнім деревом, або деревом, у якого будь-який вузол не має синів, має *лівого* або *правого сина*, має обох синів. Той факт, що кожний син будь-якого вузла визначений як лівий або як правий, суттєво відрізняє двійкове дерево від упорядкованого орієнтованого дерева.



Мал. 19.2

Структура вузла двійкового дерева зображена на малюнку 19.2. Він є структурою і містить поле (поля) для даних, а також два поля для вказівників на лівого і правого синів. У мові С тип вузла двійкового дерева можна оголосити так:

```
typedef struct elem
{       <тип даних> dani;       // поле для даних
        int kilk;               // кількість повторень даних
        struct elem * left;     // вказівник на лівого сина
        struct elem * right;    // вказівник на правого сина
}       node;
typedef node * ptr;
```

У цьому оголошенні node – тип вузла дерева, ptr – вказівник на вузол. Зробивши оголошення

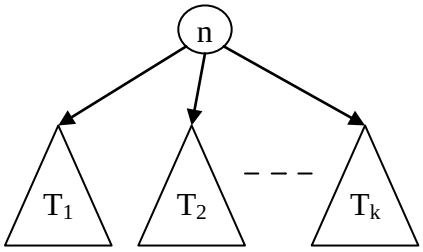
```
ptr root;
```

будемо вважати root вказівником на корін дерева.

Найпоширенішими операціями під час роботи з деревами є: створення дерева, добавлення вузла в дерево, вилучення вузла з дерева, обхід дерева та пошук у ньому.

Створення дерева полягає у добавленні до порожнього дерева певної кількості вузлів за певним правилом. Це правило визначається тим, яке дерево треба побудувати. Різновидом двійкового дерева є дерево *бінарного пошуку*. Дерева бінарного пошуку можна будувати для множин, елементи яких можна порівнювати, тобто такі множини можна упорядковувати. У дереві бінарного пошуку для будь-якого вузла *x* значення всіх вузлів лівого піддерева *x* не більші за значення *x*, а значення всіх вузлів правого піддерева *x* не менші за значення *x*.

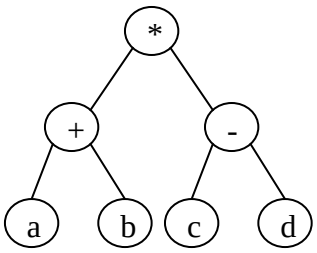
Існують різні алгоритми обходу всіх вузлів дерева. Найчастіше використовуються три способи: *обхід у прямому порядку* (зверху вниз), *обхід в оберненому порядку* (знизу вверх), *обхід у внутрішньому порядку* (симетричний обхід, зліва направо). Усі три способи обходу рекурсивно можна означити наступним чином.



Мал. 19.3

- Якщо дерево T є порожнім, то до списку обходу заноситься порожній запис.
- Якщо дерево T складається з одного вузла, то до списку обходу заноситься цей вузол.
- Далі, нехай T – дерево з коренем **n** і піддеревами T_1, T_2, …, T_k, як зображено на малюнку 19.3. Тоді для різних способів обходу маємо таке:

1. При обході в прямому порядку вузлів дерева T спочатку відвідуємо корінь **n** , потім вузли піддерева T_1, далі всі вузли піддерева T_2, і т. д. Останніми відвідуються вузли піддерева T_k.
2. При симетричному обході вузлів дерева T спочатку відвідуються в симетричному порядку всі вузли піддерева T_1, далі корінь **n**, потім послідовно в симетричному порядку всі вузли піддерев T_2, …, T_k.
3. При обході в оберненому порядку спочатку відвідуються в оберненому порядку всі вузли піддерева T_1, потім послідовно відвідуються всі вузли піддерев T_2, …, T_k також в оберненому порядку, останнім відвідується корінь n.



Мал. 19.4

На малюнку 19.4 зображене дерево виразу (a+b)*(c-d). При обході цього дерева в прямому порядку отримуємо *+ab-cd – префіксну форму виразу, а при симетричному обході – a+b*c-d – інфіксну форму, і, нарешті, при оберненому обході – ab+cd-* - постфіксну форму.

Функція обходу двійкового дерева в прямому порядку має вигляд.

```
void PreOrder(ptr root)
{if (root!=NULL)
     {      <вивести>(root->dani);
            PreOrder(root->left);
            PreOrder(root->right);
     }
}
```

Функція оберненого обходу відрізняється від наведеної вище перестановкою рядків та має вигляд.

```
void InverOrder(ptr root)
{    if (root!=NULL)
     {      InverOrder(root->left);
            InverOrder(root->right);
            <вивести>(root->dani);
     }
```

```
}
```

Аналогічно виглядає функція симетричного обходу.

```
void InOrder(ptr root)
{   if (root!=NULL)
    {       InOrder(root->left);
            <вивести>(root->dani);
            InOrder(root->right);
    }
}
```

Для двійкового дерева операція пошуку може реалізовуватися по-різному. Зокрема, для цього можна використати один із наведених вище алгоритмів обходу. Для дерева бінарного пошуку ця операція може виглядати наступним чином.

```
ptr SearchElem(ptr root, <тип даних> elem)
{   while (root!=NULL)
    {       if (root->dani==elem)                   // елемент знайдений
                    return root;
            else            // елемента не знайдено, продовжуємо пошук
                    if (root->dani > elem)
                            root = root->left;
                            else
                                    root = root->right;
    }
    return root;
}
```

У даній функції шукане значення позначене іменем elem та є параметром. При успішному пошуку функція повертає вказівник на вузол дерева зі знайденим значенням, інакше – повертає NULL.

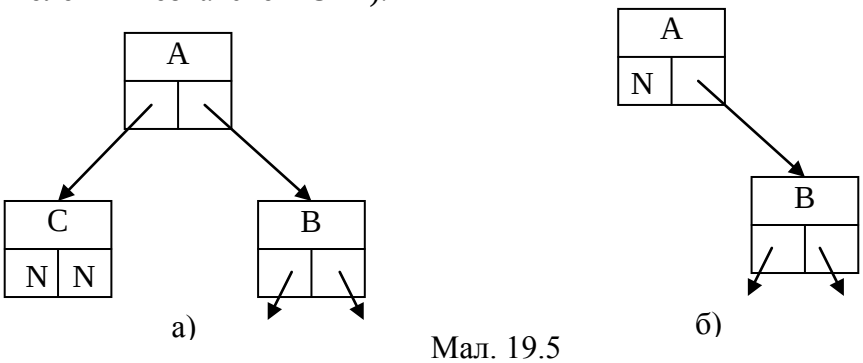
Операція добавлення вузла до дерева бінарного пошуку складається з двох частин. Спочатку відшукується місце (вузол), куди треба добавляти. Потім вже виконується саме добавлення. Можливі наступні два варіанти. Нехай елемент даних, що треба добавити до дерева, уже міститься у якомусь вузлі. Тоді, при виконанні операції добавлення вузла, у дереві новий вузол не появиться. Виконання цієї операції полягає у відшуканні вузла із потрібним значенням та збільшенні лічильника значень на 1. У другому варіанті передбачається, що елемент, який добавляється, не міститься в дереві. Тоді виконання операції полягає у пошуку відповідного листка дерева, до якого буде під'єднуватися новий вузол, що теж буде листком. Наступна функція реалізує операцію добавлення. Вона використовується для побудови дерева бінарного пошуку. Ця функція є рекурсивною, параметр proot – вказівник на root.

```
void SearchInsert(ptr * proot, <тип даних> elem)
{       if ((*proot) = NULL) // вузла з шуканим значенням elem у дереві немає, визначене
                                    місце під'єднання
        {       // виділення пам'яті під новий вузол і його заповнення
                *proot = (ptr) malloc(sizeof(node));
                (*proot)->dani = elem;
                (*proot)->kilk = 1;
                (*proot)->left = NULL;
                (*proot)->right = NULL;
        }
        else    // продовження пошуку вузла
                if (elem < (*proot)->dani)              // шукаємо в лівому піддереві
                        SearchInsert(&((*proot)->left), elem);
                else
```

```
        if (elem > (*proot)->dani)      // шукаємо в правому піддереві
                SearchInsert(&((*proot)->right), elem);
        else                    // шуканий вузол знайдений
                (*proot)->kilk++;       // збільшуємо кількість
}
```

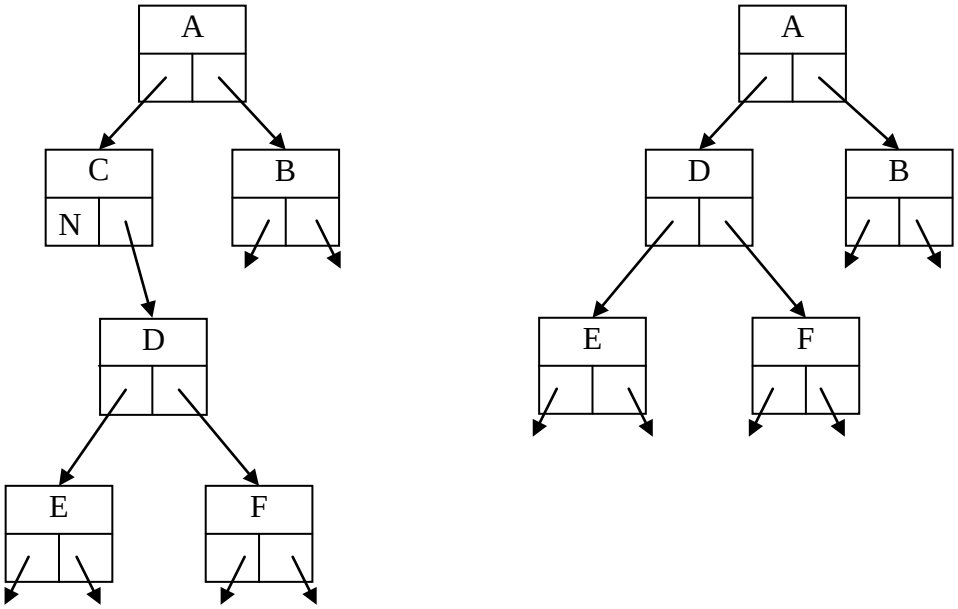
Операція вилучення вузла у дереві бінарного пошуку є найскладнішою, бо вилучення внутрішнього вузла призводить до розбиття дерева на частини, які знову необхідно об'єднати в одне дерево. Можливі три випадки для вузла, що вилучається: не має синів, тобто є листком дерева; має одного сина; має двох синів.

Найпростіше вилучити вузол, який є листком. У цьому випадку вилучаємо вузол (звільняємо від нього динамічну пам'ять), і вказівнику на цей вузол присвоюємо значення NULL. На малюнку 19.5 а) зображений фрагмент дерева до вилучення вузла C, а на 19.5 б) – після вилучення (символом N позначено NULL).



Мал. 19.5

Якщо вилучається вузол C, який має одного сина D, то вказівнику на вузол C слід присвоїти адресу D і звільнити пам'ять від C. Процес вилучення зображений на малюнку 19.6.



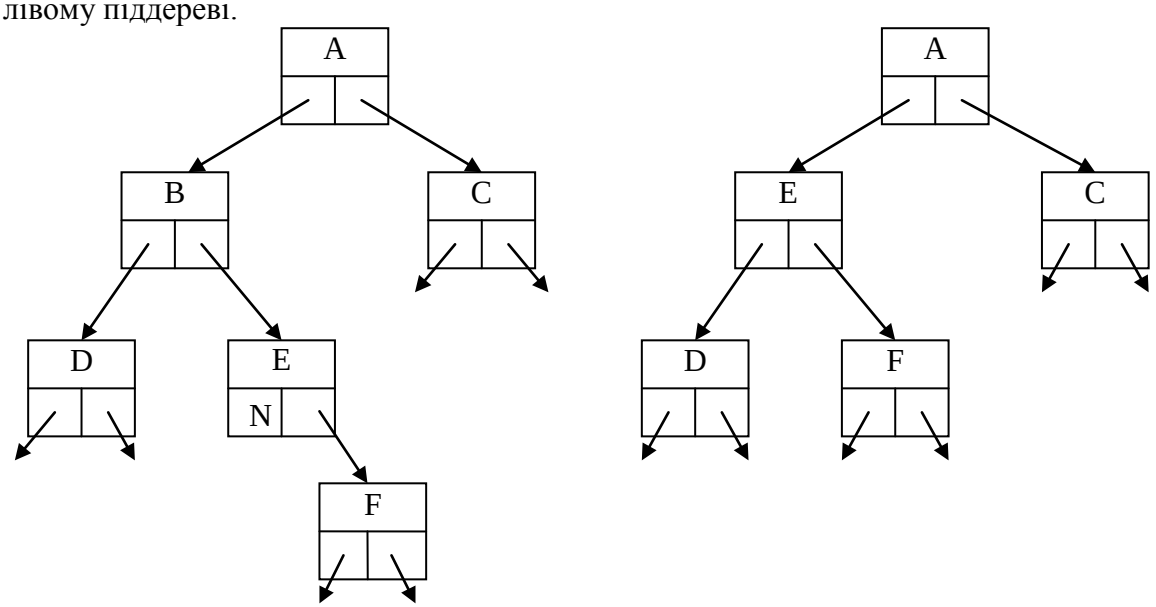
Мал. 19.6

Найскладнішим є випадок, коли вузол B, що вилучається, має двох синів, наприклад, синів D і E, як зображено нижче на малюнку 19.7. Щоб вилучити B, спочатку треба відшукати найлівіший вузол у правому піддереві B (у дереві з коренем E). Для вузла E можливі два випадки: 1) він не має лівого сина; 2) він має лівого сина.

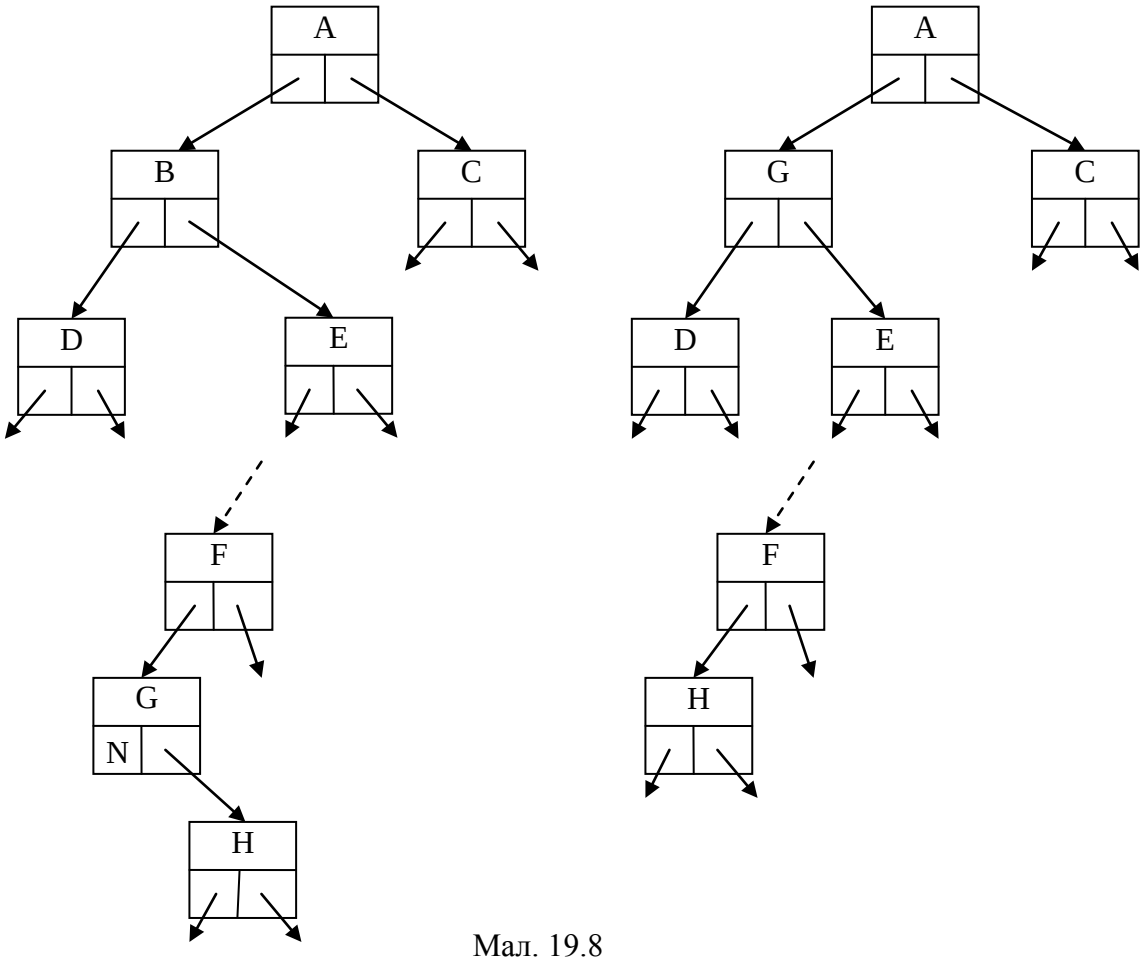
Якщо вузол E не має лівого сина, то його значення є найменшим у правому піддереві B. Тоді значення вузла B замінюємо на значення E і вилучаємо вузол E. Операція вилучення зображена на малюнку 19.7. Отримане дерево задовольняє вимогам для дерева бінарного пошуку.

Нехай тепере вузол E має лівого сина. Тоді він має найлівішого сина, який є вузлом G на малюнку 19.8. Значення G є найменшим зі значень всіх вузлів піддерева з коренем E. Процес вилучення B полягає в тому, що значення вузла G заносимо в B і вилучаємо G. В отриманому дереві відношення порядку не порушилось.

Зауважимо також, що при вилученні B можна було б шукати найправіший вузол у лівому піддереві.



Мал. 19.7



Мал. 19.8

Описаний процес вилучення вузла підходить також для випадку, коли вилучається корінь, що має двох синів. Наступна функція реалізує операцію вилучення. Вона є

рекурсивною. Через параметр proot передається вказівник на root – вказівник на корінь дерева. Також через proot у програму повертається інформація про перебудоване дерево. Параметр elem служить для передачі даних вузла, який треба вилучити.

```
void SearchDelete(ptr * proot, <тип даних> elem)
{       ptr vnode;
        ptr delnode;                    // вказівник на вузол, який треба вилучити
        if (*proot == NULL)             // дерево не містить шуканого вузла
                puts("Шуканого вузла немає, вилучити не можна");
        else    if (elem < (*proot)->dani)      // шукаємо в лівому піддереві
                        SearchDelete(&((*proot)->left), elem);
                else    if (elem > (*proot)->dani)      // шукаємо в правому піддереві
                                SearchDelete(&((*proot)->right), elem);
                        else            // шуканий вузол знайдений
        {delnode = (*proot);            // запам'ятовуємо зузол, який треба вилучати
        if (((*proot)->right = NULL) && ((*proot)->left = NULL))        // вузол не має синів
                (*proot) = NULL;        // вилучаємо вузол - листок
        else    if ((*proot)->right = NULL)             // вузол не має правого сина
                        (*proot) = (*proot)->left;      // вилучаємо вузол через під'єднання
                                                                        лівого сина
                else    if ((*proot)->left = NULL)      // вузол не має лівого сина}
                                (*proot) = (*proot)->right;     // вилучаємо вузол через
                                                                        під'єднання правого сина
                        else            // вузол має обох синів
                        {       vnode = (*proot)->right;        // правий син вузла
        `                               if (vnode->left == NULL)      // він не лівого сина
                                {(*proot)->dani = vnode->dani;        // копіюємо дані
                                delnode = vnode;                // вузол, який будемо вилучати
                                (*proot)->right = vnode->right;         // вилучення node
                                }
                                else      // правий син має лівого сина
                                {// Пошук найлівішого вузла
                                while (vnode->left->left != NULL)
                                        vnode = vnode->left;
                                delnode = vnode->left;    // найлівіший вузол, будемо вилучати
                                (*proot)->dani = delnode->dani;       // копіювання даних
                                vnode->left = delnode->right;     // вилучення найлівішого вузла
                                }
                        }
        free(delnode);          // звільнення динамічної пам'яті
        }
}
```

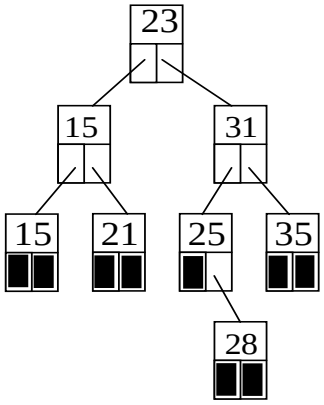
Розглянуті операції вилучення та добавлення вузлів стосувалися дерев бінарного пошуку. У програмуванні часто використовуються інші різновиди дерев, для яких ці операції можуть реалізовуватися зовсім по-іншому. Ми також не торкалися питання представлення дерев у вигляді масивів.

Приклад 1

Розглянемо приклад сортування послідовності цілих чисел з використанням дерева бінарного пошуку. Нехай послідовність чисел 23, 15, 31, 25, 21, 15, 28, 35 треба упорядкувати за спаданням.

Аналіз задачі

Створимо дерево пошуку наступним чином (мал. 19.9). Перший елемент послідовності (23) заносимо до інформаційної частини кореня (надалі слова



Мал. 19.9

"інформаційна частина" будемо опускати). Наступний елемент (15) послідовності порівнюємо з коренем Оскільки 15<23 і лівий вказівник кореня дорівнює NULL, то створюємо листок для числа 15, і цей листок під'єднуємо зліва до кореня. Аналогічно порівнюємо 31 із коренем і під'єднуємо вузол (листок) цього числа до кореня справа. Далі порівнюємо число 25 із коренем. Оскільки 25>23, то переходимо до правого сина кореня і тепер порівнюємо числа 25 та 31. Згідно нерівності 25<31, ми повинні перейти до лівого сина вузла 31. Але лівого сина немає, тому створюємо вузол для числа 25 та під'єднуємо його до вузла 31.

Отже, дерево пошуку створюється так. Черговий елемент X послідовності порівнюємо спочатку з коренем. Якщо X менший кореня, то переходимо до лівого сина, інакше до правого. Тепер X порівнюємо із сином та знову визначаємо напрямок руху по дереву. Переходячи так від вузла до вузла, ми знайдемо місце для під'єднання вузла числа X до дерева. Для будь-якого вузла дерева пошуку всі менші за значення вузла елементи знаходяться зліва, а більші – справа.

На мал. 19.9 зображене дерево з однаковими вузлами. З метою уникнення дублювання вузлів, ми будемо враховувати їх кількість. Застосувавши до дерева бінарного пошуку алгоритм симетричного обходу, отримаємо упорядковану за зростанням послідовність. Помінявши в цьому алгоритмі напрямки лівий на правий та навпаки, отримаємо спадну послідовність.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
// Визначення типу node для вузлів дерева бінарного пошуку
typedef struct elem
        {       int num;                        // Дані вузла (ціле число)
                int kilk;                       // Кількість повторень даних
                struct elem * left;             // Вказівник на лівого сина
                struct elem * right;            // Вказівник на правого сина
        }       node;
// Визначення типу ptr для вказівника на вузол дерева
typedef node * ptr;
// Функція добавляє вузол у дерево. Параметр proot є подвійним вказівником на корінь.
// Через нього передається інформація про дерево з програми у функцію та навпаки.
// Параметр elem служить для передачі числа, яке треба занести у вузол дерева.
void SearchInsert(ptr * proot, int elem);
// Функція здійснює симетричний обхід дерева і виводиться список значень вузлів у
// порядку спадання. Параметр root – вказівник на корінь.
void InOrder(ptr root);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int k;
        int i;
        ptr root = NULL;                        // Вказівник на корінь дерева
        puts("Введіть 8 чисел послідовності");
```

```
        for (i=0; i<8; i++)
        {       scanf("%d", &k);
                SearchInsert(&root, k);                 // Добавлення вузла у дерево
        }
        puts("Упорядкована послідовність");
        InOrder(root);
        printf("\n");
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------
void SearchInsert(ptr * proot, int elem)
{       // Функція добавляє вузол у дерево
        if ((*proot) == NULL)         // вузла з шуканим значенням elem у дереві немає,
                                                визначене місце під'єднання
        {       // виділення пам'яті під новий вузол і його заповнення
                *proot = (ptr) malloc(sizeof(node));
                (*proot)->num = elem;
                (*proot)->kilk = 1;
                (*proot)->left = NULL;
                (*proot)->right = NULL;
        }
        else    // продовження пошуку вузла
                if (elem < (*proot)->num)               // шукаємо в лівому піддереві
                        SearchInsert(&((*proot)->left), elem);
                else
                        if (elem > (*proot)->num)       // шукаємо в правому піддереві
                                SearchInsert(&((*proot)->right), elem);
                        else            // шуканий вузол знайдений
                                (*proot)->kilk++;       // збільшуємо кількість
}
//-------------------------------------------------------------------------------------------------------------
void InOrder(ptr root)
{       // Симетричний обхід вузлів дерева
        int i;
        if (root!=NULL)
        {       InOrder(root->right);                   // Обхід правого піддерева
                // Виведення значення вузла
                for (i=0; i<root->kilk; i++)
                        printf("%4d", root->num);
                InOrder(root->left);                    // Обхід лівого піддерева
        }
}
```

```
Введіть 8 чисел послідовності
23 15 31 25 21 15 28 35
Упорядкована послідовність
  35   31   28   25   23   21   15   15
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 2

Надрукувати всі елементи дерева по рівнях. Значення вузлів кожного рівня виводити зліва направо.

Аналіз задачі

Створимо дерево бінарного пошуку із цілих чисел за алгоритмом із прикладу 1. Виведення елементів дерева по рівнях організуємо на основі черг.

Спочатку виведемо інформацію з кореня дерева. При цьому створимо першу чергу із синів кореня, які заносимо зліва направо. Здійснюємо рух по цій черзі від голови до хвоста та на кожному кроці виконуємо дії: виводимо дані з елемента, який знаходиться в голові черги; заносимо в другу чергу синів цього елемента; вилучаємо елемент з черги. Після завершення руху по першій черзі буде створена друга черга, а перша – буде знищена. Другу чергу тепер вважаємо першою і виконуємо тепер над нею такі самі дії. Алгоритм виведення вузлів дерева припинить роботу тоді, коли на якомусь кроці друга черга буде порожньою.

Доцільно інформацію про вузол дерева виводити в такій формі, щоб на її основі можна було б візуально побудувати дерево. Кожному вузлу присвоїмо номер від 1 до n, де n – кількість вузлів. Нумерацію проводитимемо від кореня до низу дерева по рівнях у порядку виведення вузлів. Інформацію про вузол будемо подавати у вигляді п’ятірки: номер батька, сторона під’єднання до батька (L – зліва, R – справа), номер вузла, дані вузла, кількість повторень даних. На основі цієї інфомації можна побудувати дерево.

Запропонований алгоритм можна використовувати для виведення інформації про дерево в текстовому режимі роботи екрану.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
// Визначення типу node для вузлів бінарного дерева пошуку
typedef struct elemt
        {       int num;                        // Дані вузла (ціле число)
                int kilk;                       // Кількість повторень даних
                struct elemt * left;            // Вказівник на лівого сина
                struct elemt * right;           // Вказівник на правого сина
        }       node;
// Визначення типу treeptr для вказівників на вузол дерева
typedef node * treeptr;
// Визначення типу element для елементів черги
typedef struct elem
        {       int nomerbatka;                 // Номер батька даного вузла
                char storona;           // Під’єднання вузла до батька (L – зліва, R – справа)
                int nomervuzla;                 // Номер вузла
                treeptr vuzol;                  // Вказівник на вузол у дереві
                struct elem * next;
        }       element;
// Визначення типу ptr для вказівників на елементи черги
typedef element * ptr;
// Функція добавляє вузол у дерево. Параметр proot є подвійним вказівником на корінь.
// Через нього передається інформація про дерево з програми у функцію та навпаки.
// Параметр elem служить для передачі числа, яке треба занести у вузол дерева.
void SearchInsert(treeptr * proot, int elem);
// Функція виводить по рівнях дані вузлів дерева. Параметр root є вказівником на корінь.
void PrintTree(treeptr root);
// Функція добавляє елемент до черги. Параметри: phead – подвійний вказівник на
// вершину черги, ptail – подвійний вказівник на хвіст черги, nb – номер батька вузла
// дерева, st – сторона під’єднання до батька, nv – номер вузла, v – вказівник на вузол у
// дереві.
```

void InsertCherga(ptr * phead, ptr * ptail, int nb, char st, int nv, treeptr v);
// Функція вилучає елемент з черги. Параметр head – вказівник на корінь дерева.
ptr DelElementV(ptr head);

```
int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int k;
        treeptr root = NULL;                    // Вказівник на корінь дерева
        puts("Введіть цілі числа. Останне число - 111");
        scanf("%d", &k);                        // Введення числа з клавіатури
        while (k!=111)
        {       SearchInsert(&root, k);         // Занесення числа у дерево
                scanf("%d", &k);                // Введення наступного числа з клавіатури
        }
        puts("Виведення дерева по рівнях");
        PrintTree(root);                        // Виведення по рівнях значень вузлів дерева
        system("pause");
        return 0;
}
//-------------------------------------------------------------------------------------------------------------
void SearchInsert(treeptr * proot, int elem)
{       // Добавлення вузла у дерево
        if ((*proot) == NULL)           // вузла з шуканим значенням elem у дереві немає,
                                                визначене місце під'єднання
        {       // виділення пам'яті під новий вузол і його заповнення
                *proot = (treeptr) malloc(sizeof(node));
                (*proot)->num = elem;
                (*proot)->kilk = 1;
                (*proot)->left = NULL;
                (*proot)->right = NULL;
        }
        else    // продовження пошуку вузла
                if (elem < (*proot)->num)       // шукаємо в лівому піддереві
                        SearchInsert(&((*proot)->left), elem);
                else
                        if (elem > (*proot)->num)   // шукаємо в правому піддереві
                                SearchInsert(&((*proot)->right), elem);
                        else            // шуканий вузол знайдений
                                (*proot)->kilk++;   // збільшуємо кількість
}
//-------------------------------------------------------------------------------------------------------------
void PrintTree(treeptr root)
{       // Виведення вузлів дерева по рівнях
        ptr head1 = NULL;               // Вказівник на вершину першої черги
        ptr tail1 = NULL;               // Вказівник на хвіст першої черги
        ptr head2 = NULL;               // Вказівник на вершину другої черги
        ptr tail2 = NULL;               // Вказівник на хвіст другої черги
        int riven = 0;                  // Номер рівня дерева
        int i = 1;                      // Номер вузла дерева
        if (root == NULL)
                puts("Дерево порожне");
        else
```

```
        {// Виводимо дані кореня дерева
        printf("Корінь: номер - 1, число - %d, кількість - %d\n", root->num, root->kilk);
        if (root->left != NULL)                 // Корінь має лівого сина
        {       i++;                            // Номер лівого сина
                // Заносимо в першу чергу інформацію про цього сина
                InsertCherga(&head1, &tail1, 1, 'L', i, root->left);
        }
        if (root->right != NULL)                // Корінь має правого сина
        {       i++;                            // Номер правого сина
                // Заносимо в першу чергу інформацію про цього сина
                InsertCherga(&head1, &tail1, 1, 'R', i, root->right);
        }
        while (head1 != NULL)                   // Поки перша черга не порожня
        {       riven++;        // Обчислення рівня вузлів, що будуть виводитися
                printf("Вузли %d рівня\n", riven);
                while (head1 != NULL)       // Цикл для руху по першій черзі
                {// Виведення інформації про вузол, який знаходиться у вершині черги
                printf("(%d, %c, %d, %d, %d)   ", head1->nomerbatka, head1->storona,
                                head1->nomervuzla, head1->vuzol->num, head1->vuzol->kilk);
                if (head1->vuzol->left != NULL)     // Цей вузол має лівого сина
                {   i++;                            // Номер сина
                    // Заносимо в другу чергу інформацію про сина
                    InsertCherga(&head2, &tail2, head1->nomervuzla, 'L', i, head1->vuzol->left);
                }
                if (head1->vuzol->right != NULL)   // Цей вузол має правого сина
                {   i++;                            // Номер сина
                    // Заносимо в другу чергу інформацію про сина
                   InsertCherga(&head2, &tail2, head1->nomervuzla, 'R', i, head1->vuzol->right);
                }
                head1 = DelElementV(head1);         // Вилучаємо вузол з черги
                }
                printf("\n");
                head1 = head2;                      // Друга черга стає першою
                tail1 = tail2;
                head2 = NULL;                       // Друга черга порожня
                tail2 = NULL;
                }
        }
}
//-------------------------------------------------------------------------------------------------------------------
void InsertCherga(ptr * phead, ptr * ptail, int nb, char st, int nv, treeptr v)
{       // Добавлення елемента в чергу
        ptr x;
        x=(ptr) malloc(sizeof(element));
        // Занесеня даних в елемент
        x->nomerbatka = nb;
        x->storona = st;
        x->nomervuzla = nv;
        x->vuzol = v;
        x->next = NULL;
        if (*phead == NULL)
                {       *phead = x;
```

```
                *ptail = x;
        }
        else
        {       (*ptail) -> next = x;
                *ptail = x;
        }
}
//-------------------------------------------------------------------------------------------------------------------
ptr DelElementV(ptr head)
{       ptr x;
        x = head;
        head = head->next;
        free(x);
        return head;
}
```

```
Введіть цілі числа. Останне число - 111
45 22 63 -2 33 22 45 44 44
2 10 10 40 50 22 -4 70 6 55
111
Виведення дерева по рівнях
Корінь: номер - 1, число - 45, кількість - 2
Вузли 1 рівня
(1, L, 2, 22, 3)   (1, R, 3, 63, 1)
Вузли 2 рівня
(2, L, 4, -2, 1)   (2, R, 5, 33, 1)   (3, L, 6, 50, 1)   (3, R, 7, 70, 1)
Вузли 3 рівня
(4, L, 8, -4, 1)   (4, R, 9, 2, 1)   (5, R, 10, 44, 2)   (6, R, 11, 55, 1)
Вузли 4 рівня
(9, R, 12, 10, 2)   (10, L, 13, 40, 1)
Вузли 5 рівня
(12, L, 14, 6, 1)
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

Приклад 3

Задане визначення формули

<формула >::=<термінал>|(<формула><знак><формула>)

<знак>::= + | - | * | /

<термінал>::=<змінна>|<цифра>

<змінна>::=a|b|c|d|e|f|g|h|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|

<цифра>::=0|1|2|3|4|5|6|7|8|9|

Цю формулу можна представити у вигляді двійкового дерева за правилами: формула з одного терміналу (цифри або змінної) представляється деревом з одною вершиною – цим терміналом, а формула виду F1sF2 – деревом, у якому корінь – це знак s; ліве та праве піддерева – це відповідні представлення формул F1 та F2. Написати програму, яка за формулою з текстового файлу будує відповідне дерево-формулу.

Аналіз задачі

Найпростішими формулами є термінали, тобто змінна, яка складається з одної букви, або цифра. Складніші формули утворюються з використанням операцій та круглих дужок. Для побудови дерева формули створимо рекурсивну функцію, яка за рядком-формулою будує дерево. Значеннями вузлів дерева будуть символи. Функція працює наступним чином: обрив рекурсії реалізований для формули з одного символа, а для формули (<формула><знак><формула>) організовується рекурсія.

Функція будує дерево для правильної формули, а для неправильної може отриматися якесь дерево або буде виведене повідомлення про неправильну формулу. Наприклад, вираз а+в не є формулою, бо правильна формула повинна мати вигляд (а+в). Для (а+в) функція побудує правильне дерево, а для а+в дерево буде складатися з кореня а. Тому створена рекурсивна функція не проводить повного аналізу формули на правильність.

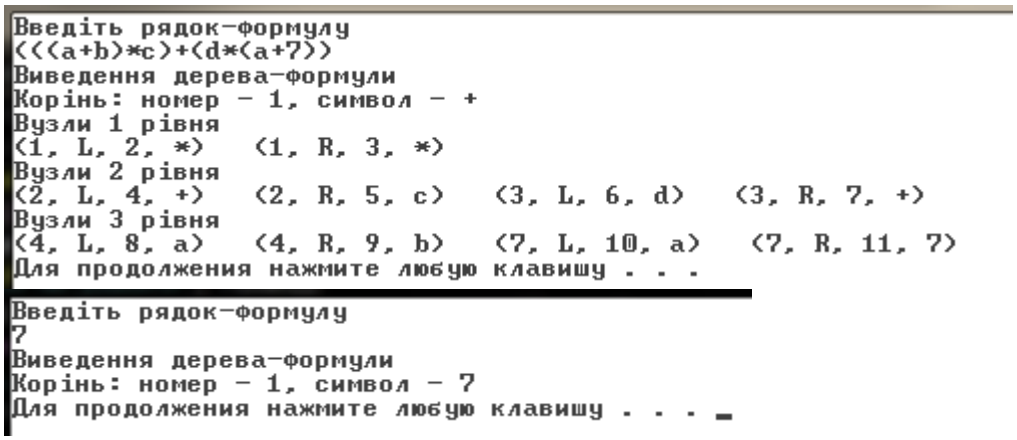
Для виведення дерева-формули використаємо функцію із прикладу 2.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "string.h"
// Визначення типу node для вузлів бінарного дерева пошуку
typedef struct elemt
        {       char sym;                       // Дані вузла (символ)
                struct elemt * left;            // Вказівник на лівого сина
                struct elemt * right;           // Вказівник на правого сина
        }       node;
// Визначення типу treeptr для вказівників на вузол дерева
typedef node * treeptr;
// Визначення типу element для елементів черги
typedef struct elem
        {       int nomerbatka;                 // Номер батька даного вузла
                char storona;           // Під'єднання вузла до батька (L – зліва, R – справа)
                int nomervuzla;                 // Номер вузла
                treeptr vuzol;                  // Вказівник на вузол у дереві
                struct elem * next;
        }       element;
// Визначення типу ptr для вказівників на елементи черги
typedef element * ptr;
// Фукнція будує дерево формули, яка задана текстовим рядком, та повертає вказівник на
// корінь побудоваго дерева.
treeptr DerevoFormula(void);
// Функція виводить по рівнях дані вузлів дерева. Параметр root є вказівником на корінь.
void PrintTree(treeptr root);
// Функція добавляє елемент у хвіст черги. Цей елемент містить інформацію про вузол
// дерева. Параметри: phead – подвійний вказівник на вершину черги, ptail – подвійний
// вказівник на хвіст черги, nb – номер батька вузла дерева, st – сторона під'єднання до
// батька, nv – номер вузла, v – вказівник на вузол.
void InsertCherga(ptr * phead, ptr * ptail, int nb, char st, int nv, treeptr v);
// Функція вилучає елемент з вершини черги та повертає вказівник на нову вершину.
// Параметр head є вказівником на вершину.
ptr DelElementV(ptr head);

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        treeptr root;                           // Вказівник на корінь дерева
        puts("Введіть рядок-формулу");
        root = DerevoFormula();                 // Побудова дерева формули
        puts("Виведення дерева-формули");
        PrintTree(root);                        // Виведення по рівнях значень вузлів дерева
        system("pause");
        return 0;
}

//-------------------------------------------------------------------------------------------------------------
treeptr DerevoFormula(void)
{       // Функція будує дерево формули
        char c, op;                             // Змінні для читання символів формули
```

```
        treeptr lf, rf, nf;                     // Вказівники на вузли дерева
        scanf("%c", &c);                        // Читаємо символ формули (рядка)
        if (((c>='a')&&(c<='z'))||((c>='0')&&(c<='9')))     // Символ належить алфавіту
        {       // Створюємо вузол дерева, який містить прочитаний символ та є листком
                nf = (treeptr)malloc(sizeof(node));         // Виділення пам’яті для вузла
                nf->sym = c;                                // Заносимо дані (символ)
                nf->left = NULL;
                nf->right = NULL;
                return nf;                      // Повертаємо вказівник на створений вузол
        }
        else            // Формула має складну структуру (F1 <операція> F2)
                if (c=='(')
                {       lf = DerevoFormula();           // Побудова дерева для формули F1
                        scanf("%c", &op);               // Читаємо символ операції
                        if ((op!='+')&&(op!='-')&&(op!='*')&&(op!='/'))
                        {       puts("Формула записана невірно");
                                exit(1);
                        }
                        rf = DerevoFormula();           // Побудова дерева для формули F2
                        // Створюємо дерево складної формули
                        nf = (treeptr)malloc(sizeof(node));   // Виділяємо пам'ять для кореня
                        nf->sym = op;                   // Заносимо символ операції
                        nf->left = lf;                  // Зліва під’єднуємо дерево формули F1
                        nf->right = rf;                 // Справа під’єднуємо дерево формули F2
                        scanf("%c", &c);                // Читаємо закриту дужку
                        return nf;      // Повертаємо вказівник на корінь побудованого дерева
                }
                else            // Формула містить символи, які не належать алфавіту
                        {       puts("Формула записана невірно");
                                exit(1);
                        }
}
//-------------------------------------------------------------------------------------------------------------
void PrintTree(treeptr root)
{       // Виведення вузлів дерева по рівнях
        ptr head1 = NULL;                       // Вказівник на вершину першої черги
        ptr tail1 = NULL;                       // Вказівник на хвіст першої черги
        ptr head2 = NULL;                       // Вказівник на вершину другої черги
        ptr tail2 = NULL;                       // Вказівник на хвіст другої черги
        int riven = 0;                          // Номер рівня дерева
        int i = 1;                              // Номер вузла дерева
        if (root == NULL)
                puts("Дерево порожне");
        else
        {// Виводимо дані кореня дерева
        printf("Корінь: номер - 1, символ - %c\n", root->sym);
        if (root->left != NULL)                 // Корінь має лівого сина
        {       i++;                            // Номер лівого сина
                // Заносимо в першу чергу інформацію про цього сина
                InsertCherga(&head1, &tail1, 1, 'L', i, root->left);
        }
        if (root->right != NULL)                // Корінь має правого сина
```

```
        {       i++;                            // Номер правого сина
                // Заносимо в першу чергу інформацію про цього сина
                InsertCherga(&head1, &tail1, 1, 'R', i, root->right);
        }
        while (head1 != NULL)                   // Поки перша черга не порожня
        {       riven++;        // Обчислення рівня вузлів, що будуть виводитися
                printf("Вузли %d рівня\n", riven);
                while (head1 != NULL)       // Цикл для руху по першій черзі
                {// Виведення інформації про вузол, який знаходиться у вершині черги
                printf("(%d, %c, %d, %c)   ", head1->nomerbatka, head1->storona,
                                                 head1->nomervuzla, head1->vuzol->sym);
                if (head1->vuzol->left != NULL)     // Цей вузол має лівого сина
                {   i++;                            // Номер сина
                    // Заносимо в другу чергу інформацію про сина
                    InsertCherga(&head2, &tail2, head1->nomervuzla, 'L', i, head1->vuzol->left);
                }
                if (head1->vuzol->right != NULL)   // Цей вузол має правого сина
                {   i++;                            // Номер сина
                    // Заносимо в другу чергу інформацію про сина
                   InsertCherga(&head2, &tail2, head1->nomervuzla, 'R', i, head1->vuzol->right);
                }
                head1 = DelElementV(head1);         // Вилучаємо вузол з черги
                }
                printf("\n");
                head1 = head2;                      // Друга черга стає першою
                tail1 = tail2;
                head2 = NULL;                       // Друга черга порожня
                tail2 = NULL;
                }
        }
}
//-------------------------------------------------------------------------------------------------------------
void InsertCherga(ptr * phead, ptr * ptail, int nb, char st, int nv, treeptr v)
{       // Добавлення елемента в чергу
        ptr x;
        x=(ptr) malloc(sizeof(element));
        // Занесеня даних в елемент
        x->nomerbatka = nb;
        x->storona = st;
        x->nomervuzla = nv;
        x->vuzol = v;
        x->next = NULL;
        if (*phead == NULL)
                {       *phead = x;
                        *ptail = x;
                }
                else
                {       (*ptail) -> next = x;
                        *ptail = x;
                }
}
//-------------------------------------------------------------------------------------------------------------
```

```
ptr DelElementV(ptr head)
{       ptr x;
        x = head;
        head = head->next;
        free(x);
        return head;
}
```



Результати роботи програми.

Задачі

1. Створити два довільних дерева (не обов'язково двійкові), використавши для представлення дерев масив із лівих синів та правих братів [3]. Побудувати із цих дерев одне дерево.
2. Удосконалити функцію побудови дерева формули з прикладу 3, яка б не тільки будувала дерево, а й перевіряла формулу на правильність.

☺ Індивідуальні завдання

Основний рівень

При розв'язуванні задач використати динамічну структуру бінарного дерева пошуку.

1. Визначити число входжень деякого елемента E до дерева T.
2. Упорядкувати послідовність цілих чисел за зростанням.
3. Обчислити середнє арифметичне всіх елементів дерева T.
4. Надрукувати елементи з усіх листків дерева T.
5. Визначити, чи входить заданий елемент до дерева.
6. Обчислити суму елементів непорожнього дерева T.
7. Знайти максимальний елемент непорожнього дерева T.
8. Знайти мінімальний елемент непорожнього дерева T.
9. Знайти всі елементи дерева, що належать деякому відрізку [a; b], де a, b – задані числа.
10. Знайти середнє квадратичне значень листків.
11. Знайти середнє геометричне елементів дерева.
12. Знайти в дереві елемент, який мінімально відрізняється від заданого.
13. Знайти в дереві елемент, який максимально відрізняється від заданого.
14. Знайти крайній правий листок непорожнього дерева T.
15. Знайти крайній лівий листок непорожнього дерева T.

Підвищений рівень

1. Формулу, яка описана нижче, можна представити у вигляді двійкового дерева за правилами: формула з одного терміналу (цифри або змінної) представляється деревом з одною вершиною – цим терміналом, а формула виду F_1sF_2 – деревом, у якому корінь – це знак s; ліве та праве піддерева – це відповідні представлення формул F_1 та F_2.
   <формула >::=<термінал>|(<формула><знак><формула>)
   <знак>::= + | - | * | /
   <термінал>::=<змінна>|<цифра>
   <змінна>::=a|b|c|d|e|f|g|h|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|
   <цифра>::=0|1|2|3|4|5|6|7|8|9|
   Написати програму, яка обчислює значення дерева-формули (значення змінних можна задати масивом).
2. Користуючись умовою задачі № 1, написати програму, яка друкує дерево-формулу у вигляді формули.
3. Користуючись умовою задачі № 1, написати програму, яка перевіряє, чи є задане двійкове дерево деревом-формулою.
4. Користуючись умовою задачі № 1, написати програму, яка спрощує задане дерево-формулу T. Спрощення полягають у заміні піддерев формул (f+0), (0+f), (f-0), (f*1), (1*f) на піддерева, що відповідають формулі f, а піддерева формул (f*0), (0*f) – на вузол з 0.
5. Користуючись умовою задачі № 1, написати програму, яка перетворює задане дерево-формулу T. Перетворення полягають у заміні піддерев формул $((f_1\pm f_2)*f_3)$ і $(f_1*(f_2\pm f_3))$ на піддерева, що відповідають формулам $((f_1*f_3)\pm(f_2*f_3))$ і $((f_1*f_2)\pm(f_1*f_3))$.
6. Користуючись умовою задачі № 1, написати програму, яка виконує перетворення обернені до перетворень у задачі № 6.
7. Користуючись умовою задачі № 1, написати програму, яка для даного дерева-формули T будує нове дерево T1, що відповідає похідній початкової формули по деякій змінній x.
8. Користуючись умовою задачі № 1, написати програму, яка за даним деревом-формулою T виводить на екран постфіксну форму формули.
9. Користуючись умовою задачі № 1, написати програму, яка за даним деревом-формулою T виводить на екран префіксну форму формули.
10. Визначити висоту дерева.
11. Порівняти два дерева на співпадання.
12. Дано число N. Побудувати дерево T за таким принципом: корінь має інформаційну частину N; другий рівень має вузли (N-1) та (N-2); третій рівень – (N-3), (N -4), (N-5), (N-6); і т.д. Останній рівень може бути неповним і має вузол з кінцевим значенням – 1.
13. Знайти довжину шляху від кореня до вузла, який містить заданий елемент.
14. Знайти в дереві вузол з найменшим значенням та вилучити його.
15. Знайти в дереві вузол з найбільшим значенням та вилучити його.

♖ Додаткові задачі

1. У зовнішньому текстовому файлі записана (без помилок) деяка програма мовою C. Відомо, що в цій програмі кожний ідентифікатор (службове слово або ім’я) містить не більше 9 латинських літер і/або цифр. Надрукувати в алфавітному порядку всі різні ідентифікатори цієї програми, вказавши для кожного з них число його входжень до тексту програми. (Врахувати, що всередині літерних значень, рядків-констант і коментарів послідовність із букв і цифр не є ідентифікатором, і що в записі дійсних чисел може зустрічатися буква E або e.) Для зберігання ідентифікаторів використати

дерево пошуку, елементами якого є пари: ідентифікатор і число його входжень до тексту програми.
2. Розв’язати задачу №1, але разом з кожним ідентифікатором друкувати у зростаючому порядку номери всіх рядків тексту програми, у яких він зустрічається.
3. Розв’язати задачу №1 за умови, що максимальна довжина ідентифікаторів наперед не відома.
4. Розв’язати задачу №2 за умови, що максимальна довжина ідентифікаторів наперед не відома.

Питання для самоконтролю

1. Як на змістовному рівні означається дерево?
2. Дайте означення упорядкованого орієнтованого дерева.
3. Дайте означення двійкового дерева.
4. Чи є двійкове дерево упорядкованим орієнтованим деревом (упорядкованим деревом, орієнтованими деревом)?
5. Чи тотожні поняття «двійкове дерево» та «дерево бінарного пошуку»? Наведіть приклади.
6. Описати тип вузла двійкового дерева.
7. Який вузол дерева називається листком дерева, а який коренем?
8. Що називається шляхом між вузлами дерева та довжиною шляху?
9. Що таке висота вузла та висота дерева?
10. Чим відрізняються алгоритми обходу двійкових дерев? Напишіть ці алгоритми.
11. Назвіть приклади використання двійкових дерев.
12. Яка максимальна кількість листків у двійковому дереві, яке має три рівні?
13. Яка максимальна кількість листків у двійковому дереві, яке має N рівнів?
14. Напишіть алгоритм пошуку значення вузла у дереві пошуку.
15. Напишіть алгоритм добавлення вузла до дерева пошуку. Які можливі ситуації при добавленні вузла?
16. Напишіть алгоритм вилучення вузла у дереві пошуку. Як змінити цей алгоритм, якщо вилучається тільки листок?

Практичне заняття № 20

*Тема: **Заголовкові файли***

Студент повинен знати: поняття заголовкового файлу та його призначення, підключення файлів до програмиі.

Теоретичні відомості

У мові С заголовкові файли є аналогами модулів. Вони не мають чіткої структури реалізації. Ім'я файлу повинно мати розширення «h». Зазвичай в заголовковому файлі користувач може розміщувати оголошення або означення констант, типів, змінних, функцій. Файл підключається до програми за допомогою директиви #include.

Приклад 1

Створити заголовковий файл, який містить функцію розв'язування квадратного рівняння.

Аналіз задачі

Текст заголовкового файлу для даної задачі має вигляд.

```
//-------------------------------------------------------------------------------------------------------------
// Заголовковий файл kvadr.h
#include "math.h"
int KvadRivn(double a, double b, double c, double *z1, double *z2)
{       // Параметри: a, b, c – коефіцієнти квадратного рівняння; z1, z2 – вказівники на
        // корені рівняння. Функція повертає кількість коренів рівняння.
        double D;
        D=b*b-4*a*c;                            // Обчислення дискримінанта
        if (D<0)                                // Дійсних коренів немає
                return 0;                       // Повертаємо кількість коренів
        else
                if (D==0)                       // Рівняння має один корінь
                {       *z1=-b/(2*a);           // Обчислюємо корінь
                        return 1;               // Повертаємо кількість корінь
                }
                else                            // Рівняння має два корені
                {       *z1=(-b+sqrt(D))/(2*a); // Обчислюємо корені
                        *z2=(-b-sqrt(D))/(2*a);
                        return 2;               // Повертаємо кількість коренів
                }
}
//-------------------------------------------------------------------------------------------------------------
```

Файл містить підключення бібліотеки стандартних математичних функцій, бо використовується функція sqrt(). Також він містить повний опис функції KvadRivn() для відшукання коренів квадратного рівняння.

Створити заголовковий файл в середовищі Visual Studio 2010 можна різними способами. Можна скористатися меню **Файл**. Через нього обрати опції: **Создать, Файл**. У діалоговому вікні «**Создать файл**» обрати шаблон «**Visual C++**» а далі «**Заголовочный файл**», "**Открыть**». Появиться робоча область, у якій можна набирати текст заголовкового файлу. Після створення файлу його потрібно зберегти в деякій папці. При підключенні файлу до програми потрібно вказати повне ім'я файлу, наприклад,

```
#include "d:\kvadr.h"
```

Краще всього створювати заголовковий файл для відкритого проекту. Для добавлення файлу до проекту можна клацнути правою кнопкою миші на папці

«**Заголовочные файлы**» в «**Обозревателе решений**» та обрати пункти меню «**Добавить**», «**Создать элемент**», «**Заголовочный файл**», ввести ім'я файлу і натиснути кнопку «**Добавить**». До проекту буде добавлений порожній файл, який далі потрібно створити. При підключенні такого файлу до програми досить вказати його ім'я.

Програма розв'язання задачі наведена.

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
#include "kvadr.h"                    // Підключення заголовкового файлу

int _tmain(int argc, _TCHAR* argv[])
{       setlocale (LC_ALL,"RUS");
        int k;                        // Кількість коренів
        double a, b, c;               // Коефіцієнти квадратного рівняння
        double x1, x2;                // Корені рівняння
        printf ("Введіть коефіцієнти квадратного многочлена\n");
        scanf("%lf%lf%lf", &a, &b, &c);
        k=KvadRivn(a, b, c, &x1, &x2);      // Виклик фукції з файлу
        if (k==0)
                printf ("Рівняння не має дійсних коренів\n");
        else
                if (k==1)
                        printf ("Рівняння має один корінь = %lf\n", x1);
                else
                        printf ("Рівняння має два корені: x1 = %lf,  x2 = %lf\n", x1, x2);

        system("pause");
        return 0;
}
```

```
Введіть коефіцієнти квадратного многочлена
1 -3 2
Рівняння має два корені: x1 = 2,000000,  x2 = 1,000000
Для продолжения нажмите любую клавишу . . .
```

Результат роботи програми.

☺ Індивідуальні завдання

Основний рівень

1. Створити заголовковий файл, що реалізує арифметику комплексних чисел. Використати його для обчислення виразу з комплексними числами.
2. Створити заголовковий файл, який реалізує обчислення об'єму, площі основи, бічну та повні поверхні циліндра. Використати його для обчислення відповідних характеристик декількох циліндрів.
3. Створити заголовковий файл, що реалізує операції додавання та віднімання квадратних матриць порядку n. Використати його для обчислення матричного виразу.
4. Створити заголовковий файл, що реалізує операцію множення матриць. Використати його для обчислення виразу A^5, де A - деяка квадратна матриця.
5. Створити заголовковий файл, що реалізує операції над векторами на площині. Використати його для обчислення площі паралелограма і його кутів, якщо паралелограм заданий на площині координатами своїх вершин.

6. Створити заголовковий файл, який реалізує обчислення об’єму, площі основи, бічної та повної поверхні конуса. Використати його для обчислення відповідних характеристик конусів.
7. Створити заголовковий файл, який реалізує обчислення площ квадрата, прямокутника, рівностороннього трикутника і круга. Використати його для обчислення площі прямокутного листа, із якого вирізали три круги, два прямокутники, чотири квадрати і один рівносторонній трикутник.
8. Створити заголовковий файл, який реалізує обчислення площ прямокутника, трикутника і трапеції, якщо відомі сторони цих фігур. Використати його для обчислення загальної площі фігур, вирізаних із прямокутного листа жесті, якщо вирізали 10 трикутників, 6 прямокутників і 5 трапецій та виміряли їхні сторони.
9. Створити заголовковий файл, що реалізує операції над векторами на площині. Використати його для обчислення площі, кутів і довжин сторін трикутнику, якщо трикутник заданий на площині координатами своїх вершин.
10. Створити заголовковий файл, що реалізує операції над векторами в просторі. Використати його для обчислення повної поверхні та об’єму чотирикутної призми, яка задана координатами своїх вершин.
11. Створити заголовковий файл, що реалізує операції дадавання та віднімання многочленів. Використати його для обчислення виразу многочленів.
12. Створити заголовковий файл, що містить функції знаходження коренів квадратного та біквадратного рівнянь. Використати його для розв’язання відповідних рівнянь.
13. Створити заголовковий файл, який містить функції наближеного обчислення визначених інтегралів методом прямокутників та трапецій. Використати його для обчислення декількох інтегралів.
14. Створити заголовковий файл, який містить підпрограми обчислення об’ємів кулі, циліндра, куба і конуса. Використати його для обчислення об’єму води, що виллється із заповненої водою циліндричної бочки, якщо у бочку закинули 3 кулі, 4 куби і 5 конусів (вважати розміри всіх фігур відомими, і закинуті у бочку фігури повністю у ній поміщаються).
15. Створити заголовковий файл, який містить підпрограми обчислення визначників матриць другого і третього порядків. Використати його для розв’язування декількох систем лінійних рівнянь відповідного порядку.

Підвищений рівень

1. Створити заголовковий файл, що реалізує операції додавання, віднімання та множення квадратних матриць порядку n. Використати модуль для обчислення виразу A^3+B*A^2-C*A, де A,B,C - матриці.
2. Створити заголовковий файл, що реалізує операції над стеком: добавлення та вилучення елементів, очищення стеку, виведення даних на екран. Продемонструвати можливості модуля на прикладі роботи зі стеком.
3. Створити заголовковий файл, що реалізує операції над чергою: добавлення та вилучення елементів, очищення черги, виведення даних на екран. Продемонструвати можливості модуля на прикладі роботи з чергою.
4. Стек реалізований у вигляді масиву. Створити заголовковий файл, що реалізує операції над стеком: добавлення та вилучення елементів, очищення стеку, виведення даних на екран. Продемонструвати можливості модуля на прикладі роботи зі стеком.
5. Черга реалізована у вигляді масиву (аналогу кільця). Створити заголовковий файл, що реалізує операції над чергою: добавлення та вилучення елементів, очищення черги, виведення даних на екран. Продемонструвати можливості модуля на прикладі роботи з чергою.

6. Створити заголовковий файл, який містить функцію побудови дерева бінарного пошуку та функції обходу вузлів дерева. Продемонструвати можливості модуля на прикладах.
7. Створити заголовковий файл, що реалізує операції додавання, віднімання та множення на сталий множник многочленів. Продемонструвати можливості модуля на прикладах.
8. Створити заголовковий файл, що містить алгоритми сортування масиву та файлу: метод бульбашок, швидке сортування. Продемонструвати можливості модуля на прикладах.
9. Створити заголовковий файл, що містить алгоритми пірамідального сортування масиву та файлу. Продемонструвати можливості модуля на прикладах.
10. Створити заголовковий файл, що реалізує операції над векторами в тривимірному просторі. Використати модуль для обчислення об'єму паралелепіпеда, площ та кутів його граней.
11. Створити заголовковий файл, що містить підпрограми сортування елементів стеку та пошуку мінімального і максимального елементів. Продемонструвати можливості модуля на прикладі конкретного стеку.
12. Створити заголовковий файл, що містить підпрограми сортування елементів черги та пошуку мінімального і максимального елементів. Продемонструвати можливості модуля на прикладі конкретної черги.
13. Створити заголовковий файл, який містить алгоритми відшукання НСД та НСК двох чисел. Використати модуль для обчислення НСД та НСК чотирьох натуральних чисел.
14. Створити заголовковий файл, що містить підпрограми пошуку елементів у масиві: бінарний пошук, пошук всіх мінімальних (максимальних) елементів, пошук всіх елементів, рівних даному. Продемонструвати можливості модуля на прикладі конкретного масиву.
15. Створити заголовковий файл, що містить підпрограми пошуку елементів у файлі: бінарний пошук, пошук всіх мінімальних (максимальних) елементів, пошук всіх елементів, рівних даному. Продемонструвати можливості модуля на прикладі конкретного файлу.

☝ Питання для самоконтролю

1. Призначення заголовкових файлів.
2. Директива #include. Її призначення та варіанти використання.
3. Директиви макропідстановок та їх призначення.
4. Макроси з параметрами.
5. Директиви умовної компіляції. Для чого потрібна така компіляція?
6. Стандартні макроси.
7. Директива #pragma.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абрамов С.А., Гнєздилова Г.Г., Капустина Е.Н., Селюн М.И. Задачи по программированию. – М.: Наука, 1988.
2. Абрамов С.А., Зима Е.В. Начала информатики. – М.: Наука, 1989.
3. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы.–М.: Изд.дом «Вильямс», 2001.
4. Вирт Н. Алгоритмы +структуры данных = программы. – М.: Мир, 1989.
5. Дейтел Х., Дейтел П. Как программировать на С.-М.: Бином – Пресс, 2002.
6. Зеленяк О.П. Практикум программирования на Turbo Pascal. Задачи, алгоритмы и решения. – К.: ДиаСофт, 2001.
7. Зубенко В.В. Омельчук Л.Л. Програмування: навчальний посібник. К.: Видавничо-поліграфічний центр «Київський університет», 2011.
8. Ковалюк Т.В. – Основи програмування. – К.: Видавнича група BHV, 2005.
9. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ. – М.: МИНМО. 2001.
10. Коффман Э.Б. Turbo Pascal. – М.: Издательский дом «Вильямс», 2002.
11. Паращук С.Д., Троценко З.М. Програмування мовою Паскаль. Лабораторний практикум. Навчальний посібник. – Кіровоград: ПП «Центр оперативної поліграфії «Авангард», 2008.
12. Пильщиков В.Н. Сборник упражнений по язику Паскаль. –М.: Наука, 1989.
13. Подбельский В.В., Фомин С.С. Курс программирования на языке Си: учебник. – М.: ДМК Пресс, 2012.
14. Прата Стівен. Язык программирования С. Лекции и упражнения. СПб.:ООО «ДиаСофт ЮП», 2002.
15. Прата Стівен. Язык программирования С++. Лекции и упражнения.- М.: Изд.дом «Вильямс», 2011.
16. Фридман А., Кландер Л., Михаэлис М., Шильдт Х. С / С++. Архив программ. – М.: ЗАО «Издательство БИНОМ», 2001.
17. Хєзфильд Р., Кирби Л. и др. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений. Энциклопедия программиста. – К.: Издательство «Диа Софт» 2001.
18. Шпак З.Я. Програмування мовою С. – Львів: Оріяна-Нова, 2006.

НОТАТКИ

ПРАКТИКУМ
із ПРОЦЕДУРНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ
(мова С)

Навчальний посібник для студентів вищих закладів освіти

Паращук Степан Дмитрович

Формат 60x84 $^{1}/_{8}$. Папір офсет. Друк різограф.
Ум. друк. арк. 25,6. Тираж 100. Зам. 5962.

ФО-П Александрова Марина В'ячеславівна.
Свідоцтво про реєстрацію серія БО 2 №521706.
м. Кіровоград, вул. Пашутінська, 12, оф. 4
тел./факс: (0522) 24-86-34, 27-02-24,
www.avangard.kr.ua, e-mail: info@avangard.kr.ua